The Journal of Logic and Algebraic Programming 78 (2009) 117–137

Contents lists available at ScienceDirect

The Journal of Logic and Algebraic Programming

journal homepage: www.elsevier.com/locate/jlap

Processes with local and global liveness requirements

Hélia Guerra [a,*], José Félix Costa [b,c]

[a] Department of Mathematics, University of Azores, R. Mãe de Deus, 58, 9501-801 Ponta Delgada, Portugal
[b] Department of Mathematics, Technical University of Lisbon, Av. Rovisco Pais, 1049-001 Lisboa, Portugal
[c] Centro de Matemática e Aplicações Fundamentais, University of Lisbon, Lisboa, Portugal

ARTICLE INFO

Article history:
Received 10 October 2006
Revised 22 February 2008
Accepted 25 June 2008
Available online 2 October 2008

Keywords:
Process algebra
Liveness
Quiescence
Inequational proof system
Operational semantics
Conservative extension

ABSTRACT

The deterministic QS model, introduced in Costa and Sernadas [J.F. Costa, A. Sernadas, Progress assumption in concurrent systems, Formal Aspects Comput. 7 (1) (1995) 18–36], captures (local) liveness properties, commonly specified in Temporal Logic, and not fully captured by non-deterministic process models. Liveness explains how some processes engage spontaneously in some actions and wait passively for the triggering of other actions by other processes. In this paper, we extend the QS model to describe liveness properties, through the introduction of a new operator deeply influenced by Temporal Logic, and denoting a global liveness requirement. The new operator applied to a process term induces a transactional behaviour until the execution of some specified action. We define the suitable denotational, axiomatic, and operational semantic domains to obtain a trinity of semantics in the sense of Hennessy. We prove that this extended model is a conservative extension of the previous one.

1. Introduction

Liveness properties of concurrent systems [26] are commonly specified by temporal formulas, and several suitable semantic models have been proposed to these logical specifications [18]. Many of these properties are not captured by well-known non-deterministic process models, such as the input/output systems of Jonsson [14,15], failure extensions for the CSP language [25,29], and the input/output automata of Lynch [16,17]. However, in the Quiescence Model (QS model) proposed in [6,7] for sequential deterministic processes, new insights on quiescence were given concerning the characterization of input/output systems of Jonsson in [14,15]. The QS model is able to describe liveness requirements concerning the capability of processes to engage spontaneously in some actions, and to wait passively for triggering by other processes before engaging in some nonspontaneous action. This difference is captured by the introduction of two distinct prefixing operators in the language: active and passive prefixing operators. The active prefixing is also related with the strong prefixing introduced in [9], for extending CCS with atomic actions. There the prefixing operation is assembled with an underlined action meaning that the process can perform the action only as the beginning of an atomic sequence, which ends with a nonunderlied action. The QS model in [6,7], equipped with a strong prefixing operator, is also effective to explain how to look at processes with transactions, whereas strong prefixing in [9] is not used to explain transactional behaviour.

The idea of quiescence, due to Chandy and Misra [22,23,21], was developed by Jonsson [14,15], in the context of a logic to reason about specifications of input/output systems, and also by Sernadas, Ehrich, and Costa in [5,30,6,7] within an algebraic framework, in order to provide a solution for the problem of representing state-dependent liveness as a process.

The basic idea of QS model was throw away the traditional prefix-closure assumption in the classical trace semantic models, such as [12,29,4]. A QS process is in a quiescent state iff it is not willing to perform any action. So, when a QS process

* Corresponding author. Tel.: +351 296 650 509; fax: +351 296 650 072.
E-mail addresses: helia@uac.pt (H. Guerra), fgc@math.ist.utl.pt (J.F. Costa).

1567-8326/$ - see front matter © 2008 Elsevier Inc. All rights reserved.
doi:10.1016/j.jlap.2008.08.003

is in a non-quiescent state it means that some action must be performed. According to the QS model, a process (or a process community) is represented by the set of its quiescent traces, *i.e.*, the set of traces obtained in a state after that there is no commitment for the process to perform any further action. The degree of liveness associated to each process makes the corresponding set of quiescent traces, *i.e.*, the quiescent set, non prefix-closed. Two QS processes with the same prefix-closure of their traces can be comparable with respect to liveness: the smaller number of quiescent traces corresponds to the higher degree of liveness. Moreover, a trace of a QS process community is quiescent iff it can be projected onto a quiescent trace of every one of its components, because the whole community is in a quiescent state iff all its components are in a quiescent state.

Consider the well-known Hoare's vending-machine in [12] appealing to the liveness differences just described. Let $machine_1$ be a process that describes the behaviour of a vending-machine that repeatedly is compelled to provide either *coffee* or *tea* after receiving a *coin* from a customer. The customer takes the initiative to insert the *coin* whereas the vending-machine waits passively for it. However, the vending-machine takes the initiative to provide *coffee* or *tea*, after the insertion of a *coin*, and the customer waits passively for *coffee* or *tea*. The corresponding syntax in a CPS notation [12] is

$$machine_1 = \mu x.coin.(coffee.x + tea.x).$$

The standard set of traces for $machine_1$ is the regular language $(coin\,coffee + coin\,tea)^*(\epsilon + coin)$. Due to the fact that the set of traces is prefix-closed, it does not provide any information about the process liveness. The process is always in a quiescent state. However, we want that after receiving coin, the process must be in a non quiescent state eager to make *coffee* or *tea* happen. With the QS model, it is possible to express this process liveness requirement. The QS process term

$$machine_2 = \mu x.coin.(!coffee.x + !tea.x),$$

resulting from the process term $machine_1$, after to add the ! indicating the process liveness, is suitable to describe the behaviour of the vending machine. Now, the set of traces for $machine_2$ is the non prefix-closed regular language $(coin\,coffee + coin\,tea)^*$. It indicates that after *coin* the process is in a non-quiescent state, willing to perform either *coffee* or *tea*.

A denotational trace semantics and a sound and complete inequational proof system for processes with local liveness requirements were defined in the QS model in [5,7]. However, there is no operational semantics. So, we must start to provide an operational semantics to capture the implementation flavour of this process algebra. It is well known that structural operational semantics [27,28] is the dominant way of providing an operational semantics model for process algebra [12,20,11,3,2]. However, these operational semantic models do not fit well with the new liveness requirements of QS model. In our opinion, a finite automata [13] based operational semantics, such as in [4], arises more naturally than the previously mentioned operational semantics, mainly due to the fact that we can easily identify every term denoting a process in a non-quiescent state with a non-final state and every term denoting a process in a quiescent state with a final state. We continue having the transition relation defined by SOS style rules [27,28]. Moreover, this new operational semantics gives a means to complete Hennessy's trinity (equivalences among the three semantic domains), since the equivalence between this automata semantics and the denotational trace semantics can be obtained by identifying, for each closed process term, the language recognized by the corresponding automata with the respective set of traces.

In this paper we also extend the QS model in order to describe liveness properties through the introduction of a new operator in the corresponding language signature, that is deeply influenced by the (future) temporal eventually operator $\Diamond$. This provides a means to incorporate temporal operators directly in the process algebra, by contrast with the most well-known process algebras [12,19,11,3,2], where liveness properties are specified by logic formulas and verified by a model checker or by a formal proof method. This idea of incorporate temporal operators in a process algebra can be a starting point to establish a bridge from temporal logic to process algebra.

In this QS extended model, liveness requirements that induce future (eventually non-immediate) performances are called *global liveness requirements*. The liveness requirements introduced in the QS model are called *local liveness requirements*.

We enrich the QS model with an operator $[-]$ denoting a global liveness requirement. The operator $[-]$ applied to a process term induces a transactional behaviour until the execution of some arbitrary action. When we apply the global operator $[coffee]$ to $machine_1$ or $machine_2$, we restrict the corresponding sets of traces to sets, where each trace contains at least one occurrence of *coffee*, which correspond, for both processes, to the non prefix-closed regular language $(coin\,tea)^*(coin\,coffee)(coin\,coffee + coin\,tea)^*$. Notice that the global operator induces the transactional behaviour *coin coffee* for both processes.

The idea of global liveness requirements was first introduced in [5]. There an operator denoting the maximum activity of a process was proposed in an algebraic theory (with no parallel composition) of finite processes.

For the new model, called QS_{box}, an extension of the QS model, we are going to define the suitable denotational, axiomatic, and automata semantic domains. Denotational and axiomatic semantics are adapted from the corresponding semantics of the QS model, by adding for the global operator suitable sets of traces and axioms, respectively. In automata semantics, the transition relation includes suitable SOS style rules for the global operator. We found that this global operator induces an equivalence relation between process terms. So, instead of having each term denoted by a state of an automaton, as for QS automata semantics, we have each state of an automaton denoting an equivalence class of terms. We prove the equivalence among the three semantic domains. The equivalence between denotational and automata semantics is obtained through the automata languages. We also prove that the axiomatic semantic domain is a conservative extension of the corresponding QS

semantic domain. By saying that a semantic domain is conservative, we mean that it does not affect the semantic of terms over the original signature (see *e.g.* [8,1,33]).

The rest of this paper is organized as follows. In Section 2 we revisit the QS model [7], including its language for the process terms and the denotational and axiomatic domains, and then we provide a equivalent automata semantics. In Section 3 we present the extension of the QS model with the global operator [–], providing a language for process terms with global liveness requirements, together with suitable denotational, axiomatic, and automata semantic domains. Following Hennessy's methodology, we prove the equivalence among the three semantic domains and also that the model is a conservative extension of QS model. Section 4 is dedicated to conclusions and further work. Some proofs have been moved to Appendix.

2. The QS model

This section is dedicated to QS processes and is organized as follows: first we introduce the envisaged process language; next, the corresponding denotational and axiomatic semantic domains defined in [7] are briefly presented and, finally, we provide an operational semantics equivalent to the denotational and the axiomatic semantics. The last subsection is dedicated to the proofs of the required equivalences.

2.1. The syntax of QS processes

Given a finite alphabet Act of actions, the signature for the QS model is defined as follows, where $\mathcal{P}_{ne}(Act)$ denotes the nonempty subsets of Act.

Definition 1. The signature for the QS model is $\Sigma_{QS} = \Sigma^0_{QS} \cup \Sigma^1_{QS} \cup \Sigma^2_{QS}$, where

$\Sigma^0_{QS} = \{abort_U : U \in \mathcal{P}_{ne}(Act)\} \cup \{stop_U : U \in \mathcal{P}_{ne}(Act)\}$,

$\Sigma^1_{QS} = \{!a. : a \in Act\} \cup \{a. : a \in Act\}$ (active prefixing, passive prefixing),

$\Sigma^2_{QS} = \{+, \|\}$ (choice, parallel composition).

Each Σ^j_{QS}, for $j = 0, \ldots, 2$ is the set of the j-ary operation symbols. As usual, we write $!a.p$ and $a.p$ instead of $!a.(p)$ and $a.(p)$, respectively. We also write the binary symbols using infix notation. We assume that prefixing has priority over parallel composition, that has priority over choice. Intuitively, for each alphabet U, the process terms $abort_U$ and $stop_U$ represent two processes that perform no actions. The former indicates a deadlocked process, whereas the latter indicates an aborted process. The process term $!a.p$ denotes a process that first must engage in a and then behaves like p, whereas $a.p$ denotes a process that first may engage in a and then behaves like p. The process term $p + q$ behaves like p or q, depending on whether the first action is one of p or one of q. If the first action is common to p and q, $p + q$ behaves like both p and q, otherwise $p + q$ behaves like either p or q. The process term $p\|q$ denotes the combination of interleaving and synchronization of p and q. They must synchronize on all actions in the intersection of their alphabets.

Let $\Sigma^\dagger_{QS}$ denotes $\Sigma_{QS}\backslash\{\|\}$. Let $(X_U)_{U \in \mathcal{P}_{ne}(Act)}$ be a collection of infinite countable totally ordered sets (of process variables) which are pairwise disjoint and disjoint from Act. We also let x_U, y_U, and z_U range over X_U, and x, y, and z range over X.

Definition 2. The set of recursive terms over Σ_{QS}, denoted by $Rec_{QS}(X)$, is the least set which satisfies the following conditions:

1. if $x \in X$, then $x \in Rec_{QS}(X)$;
2. if $f \in \Sigma^k_{QS}$ and $t_1, \ldots, t_k$ are in $Rec_{QS}(X)$, then $f(t_1, \ldots, t_k) \in Rec_{QS}(X)$, with $k = 0, 1, 2$;
3. if $x \in X$ and $t \in Rec_{QS}(X)$, then $\mu x.t \in Rec_{QS}(X)$.

A process term defined as in (1) or (2) is called (syntactically) finite. Let $Rec^\dagger_{QS}(X)$ denote the set of recursive terms over $\Sigma^\dagger_{QS}$. To every term we assign an alphabet, that is the set of actions in which the corresponding process may engage, as follows.

Definition 3. The alphabet $\alpha : Rec_{QS}(X) \to \mathcal{P}_{ne}(Act)$ is defined inductively as follows:

1. $\alpha(abort_U) = \alpha(stop_U) = \alpha(x_U) = U$;
2. $\alpha(!a.p) = \alpha(a.p) = \{a\} \cup \alpha(p)$;
3. $\alpha(p + q) = \alpha(p\|q) = \alpha(p) \cup \alpha(q)$;
4. $\alpha(\mu x.p) = \alpha(x) \cup \alpha(p)$.

Given a recursive term p, the set of subterms of p and the set of free variables of p are given as follows.

Definition 4. For $p \in Rec_{QS}(X)$, the set of subterms of p, denoted by $Sub_{QS}(p)$, is defined inductively as follows:

1. p is a subterm of p;
2. if p is $!a.q$ or $a.q$, then every subterm of q is a subterm of p;

3. if p is $p_1 + p_2$ or $p_1 \| p_2$, then every subterm of p_1 and every subterm of p_2 is a subterm of p;
4. if p is $\mu x.q$, then every subterm of q is a subterm of p.

Definition 5. Let p be a term in $Rec_{QS}(X)$. The set of free variables of p, denoted by $\phi(p)$, is defined inductively as follows:
1. $\phi(x) = \{x\}$;
2. $\phi(abort_U) = \phi(stop_U) = \emptyset$;
3. $\phi(!a.q) = \phi(a.q) = \phi(q)$;
4. $\phi(q + r) = \phi(q \| r) = \phi(q) \cup \phi(r)$;
5. $\phi(\mu x.q) = \phi(q) \backslash \{x\}$.

A variable x occurs free in a term p if $x \in \phi(p)$. Otherwise x occurs bound. We say that $p \in Rec_{QS}(X)$ is closed if $\phi(p) = \emptyset$.

Definition 6. A term $p \in Rec_{QS}(X)$ is called guarded if in every recursive subterm $\mu x.q$ of p, every free occurrence of x in q occurs within a subterm of the form $!a.r$ or $a.r$ in q.

Definition 7. A term p is called passively guarded if in every recursive subterm $\mu x.q$ of p, every free occurrence of x in q occurs within a subterm of the form $a.r$ in q.

Now we are able to introduce the language of processes with local liveness requirements.

Definition 8. The set of process terms with local liveness requirements, denoted by $Proc_{QS}(X)$, is the subset of $Rec_{QS}(X)$, where each term p satisfies the following conditions:
1. it is passively guarded;
2. every subterm $!a.q$ or $a.q$ of p satisfies $a \in \alpha(q)$;
3. every subterm $q + r$ of p satisfies $\alpha(q) = \alpha(r)$;
4. every subterm $\mu x.q$ of p satisfies $\alpha(x) = \alpha(q)$.

As a consequence of the last definition, and assuming that $tick \in \alpha(x)$, the terms $\mu x.x$ and $\mu x.!tick.x$ are not process terms in $Proc_{QS}(X)$ because the former is not guarded and the latter, even though it is guarded, it is not passively guarded. The terms $\mu x.tick.x$ and $\mu x.coin.!choc.x$ are process terms in $Proc_{QS}(X)$ because both are passively guarded.

Let $fProc_{QS}(X)$ and $cProc_{QS}(X)$ denote, respectively, the set of all syntactically finite process terms and the set of all closed process terms. Let $cfProc_{QS}(X)$ denotes the set of all closed finite process terms. Let $Proc^{\dagger}_{QS}(X)$ denotes the subset of $Proc_{QS}(X)$ of process terms also in $Rec^{\dagger}_{QS}(X)$ (whenever necessary we also use the prefix notation c and f).

We are also interested in process terms in $cProc_{QS}(X)$ with some restrictions related to the *menu* of actions in which every process subterm may engage initially. The main reason is concerned with the operational semantics in subsection 2.4, more precisely, in obtaining directly from process terms deterministic automata. We first define the *menu* function, and then the restricted set of process terms.

Definition 9. Let p be a term in $cProc_{QS}(X)$. The *menu* of actions of p, denoted by $\nu(p)$, is defined inductively as follows:
1. $\nu(abort_U) = \nu(stop_U) = \emptyset$;
2. $\nu(!a.q) = \nu(a.q) = \{a\}$;
3. if $\nu(q)$ and $\nu(r)$ are defined, then $\nu(q + r) = \nu(q \| r) = \nu(q) \cup \nu(r)$;
4. if $\nu(q)$ is defined, then $\nu(\mu x.q) = \nu(q)$.

Definition 10. The set of process terms denoted by $cProc'_{QS}(X)$, is the subset of $cProc_{QS}(X)$, where for each term p, every subterm $q + r$ of p satisfies $\nu(q) \cap \nu(r) = \emptyset$.

Although the menu function is not defined for all process terms it does not compromise the set of process terms that we are interested (*vide* Definition 10) which applies only to closed process terms satisfying Definition 8.

2.2. The denotational semantics of QS processes

We impose an algebraic complete partial order[1] for the process terms by choosing a Σ_{QS}-domain (see [24]) $A = \langle |A|, \leq_A, \Sigma_A \rangle$, where $|A|$ is the carrier set (its elements are called A-processes) such that, for every alphabet $U \in \mathcal{P}_{ne}(Act)$, there is a pair $\langle |A_U|, \leq_A \rangle$, with $|A_U| \subseteq |A|$, which is an algebraic complete partial order under $\leq_A$, and Σ_A provides for each n-ary symbol

[1] By a complete partial order (cpo) [31] we mean a partial order $A = \langle |A|, \leq_A \rangle$ with a least element $\perp_A$ and such that every directed subset D of elements of A has a least upper bound denoted by $\bigsqcup_A D$. An element $a \in A$ is compact or finite if whenever $a \leq_A \bigsqcup_A D$, with D a direct subset of A, there exists some $d \in D$ such that $a \leq_A d$. A is an algebraic cpo if for every $a \in A$ $a = \bigsqcup_A \{d \leq_A a : d \text{ is compact}\}$.

f of Σ_{QS} a n-ary continuous function f_A within $|A|$. Moreover, we need a set ENV of environments consisting of mappings $\rho : X \to A$ that respect alphabets, *i.e.*, $\alpha(x) = U$ implies $\rho(x) \in A_U$. Then the denotation of a term p, in a given environment ρ, is taken to be $A[\![p]\!]\rho$, where $A[\![]\!]$ is defined as follows: (a) $A[\![x]\!]\rho = \rho(x)$, (b) $A[\![f(t_1,\ldots,t_m)]\!]\rho = f_A(A[\![t_1]\!]\rho,\ldots,A[\![t_m]\!]\rho)$, and (c) $A[\![\mu x.p]\!]\rho = \gamma_{\Phi_{p,\rho}}$ is the least fixed point of $\lambda\xi.A[\![p]\!]\rho[\xi/x])$, where $\rho[\xi/x]$ is the modified environment that agrees with ρ, except for the variable x whose value is ξ. As expected, with respect to such a model A, two process terms p and q are said to be equal if they denote the same A-process, *i.e.*, $A[\![p]\!]\rho = A[\![q]\!]\rho$, for every environment ρ. Moreover, p is said to be more spontaneous then q, with respect to A, if $A[\![p]\!]\rho \leq_A A[\![q]\!]\rho$, for every environment ρ. That is, $\leq_A$ partially orders the A-processes according to their liveness. If, for every environment ρ, $A[\![p]\!]\rho \leq_A A[\![q]\!]\rho$, then we write $\models_A p \leq q$.

An important aspect of the denotational semantics is given by the *finite approximation theorem* (see, *e.g.* [11]), which states that the denotational semantics of a process term is the least upper bound of the denotational semantics of its finite approximations. A finite approximation is given as follows:

Definition 11. Let $p \in Proc_{QS}(X)$ and $n \geq 0$. The *nth finite approximation of p*, denoted by p^n, is defined inductively as follows:

1. $p^0 = abort_{\alpha(p)}$;
2. $x_U^{n+1} = x_U$
3. $f(t_1,\ldots,t_k)^{n+1} = f(t_1^{n+1},\ldots,t_k^{n+1})$, with $f \in \Sigma_{QS}^k$, for $k = 0,1,2$;
4. $(\mu x.t)^{n+1} = t^{n+1}[(\mu x.t)^n/x]$

Given $p \in Proc_{QS}(X)$, let $App(p) = \{p^n : n \geq 0\}$. Before present the denotational semantics, it is useful to recall the familiar projection operation on traces, and also to define the weaving or shuffling operation of traces.

Definition 12. Let $U \subseteq Act$ and $s \in Act^*$. The *projection of s on U*, denoted by $s \downarrow U$, is defined inductively as follows:

1. $\epsilon \downarrow U = \epsilon$;
2. $s'a \downarrow U = s' \downarrow U$, $a \notin U$;
3. $s'a \downarrow U = (s' \downarrow U)a$, $a \in U$.

Definition 13. Let S and T be two sets of traces over U^* and V^*, respectively. The *weaving* (or *shuffling*) of S and T, denoted by $A^U \|^V B$, is the set $\{s \in (U \cup V)^* : s \downarrow U \in S \text{ and } s \downarrow V \in T\}$.

Let $L \subseteq U^*$ and $a \in Act$. From now on, we simply write $a.L$ instead of $\{as : s \in L\}$. The envisaged Σ_{QS} – domain is easy established.

Definition 14. The algebra of process terms with local liveness requirements is $\mathbb{QS} = \langle |\mathbb{QS}|, \leq_{\mathbb{QS}}, \Sigma_{\mathbb{QS}}\rangle$, where

- $|\mathbb{QS}| = \{\langle U, L\rangle : U \in \mathcal{P}_{ne}(Act) \text{ and } L \subseteq U^*_\downarrow\}$
- $\langle U, L\rangle \leq_{\mathbb{QS}} \langle U', L'\rangle$ iff $U = U'$ and $L \subseteq L'$
- $\Sigma_{\mathbb{QS}} = \{abort^U_{\mathbb{QS}}, stop^U_{\mathbb{QS}} : U \in \mathcal{P}_{ne}(Act)\} \cup \{!a._{\mathbb{QS}}, a._{\mathbb{QS}} : a \in Act\} \cup \{+_{\mathbb{QS}}, \|_{QS}\}$, where

$$
\begin{array}{rl@{\qquad}rl}
abort^U_{\mathbb{QS}} : & \to |\mathbb{QS}| & abort^U_{\mathbb{QS}} &= \langle U, \emptyset\rangle \\
stop^U_{\mathbb{QS}} : & \to |\mathbb{QS}| & stop^U_{\mathbb{QS}} &= \langle U, \{\epsilon\}\rangle \\
!a._{\mathbb{QS}} : & |\mathbb{QS}| \to |\mathbb{QS}| & !a._{\mathbb{QS}}\langle U, L\rangle &= \langle \{a\} \cup U, a.L\rangle \\
a._{\mathbb{QS}} : & |\mathbb{QS}| \to |\mathbb{QS}| & a._{\mathbb{QS}}\langle U, L\rangle &= \langle \{a\} \cup U, \{\epsilon\} \cup a.L\rangle \\
+_{\mathbb{QS}} : & |\mathbb{QS}| \times |\mathbb{QS}| \to |\mathbb{QS}| & \langle U, L\rangle +_{QS} \langle U', L'\rangle &= \langle U \cup U', L \cup L'\rangle \\
\|_{\mathbb{QS}} : & |\mathbb{QS}| \times |\mathbb{QS}| \to |\mathbb{QS}| & \langle U, L\rangle \|_{\mathbb{QS}} \langle U', L'\rangle &= \langle U \cup U', L^U \|^{U'} L'\rangle
\end{array}
$$

Given a $\mathbb{QS}$-process $\langle U, L\rangle$ we refer to U as $\alpha(\langle U, L\rangle)$, *i.e.*, the alphabet of $\langle U, L\rangle$, and to L as $\tau(\langle U, L\rangle)$, *i.e.*, the quiescent traces of $\langle U, L\rangle$. The proof that $\langle |\mathbb{QS}|, \leq_{\mathbb{QS}}, \Sigma_{\mathbb{QS}}\rangle$ is a Σ_{QS}-domain, where $\langle |\mathbb{QS}|, \leq_{\mathbb{QS}}\rangle$ is an algebraic cpo can be found in [7]. The following result may also be found in [7].

Proposition 15. *The Σ_{QS} – domain is finitary, i.e., every closed (sintactically) finite term is interpreted in $|\mathbb{QS}|$ as a finite element.*

2.3. The axiomatic semantics of QS processes

We now want to give a syntactical characterization of the QS model through a suitable sound and complete inequational inference system. But, first, we need to recall a few concepts. As usual, an inequation with variables in X is of the form $p \leq q$, where p and q are terms in $Rec_{QS}(X)$. Given a set In of inequations (said the *proper axioms*), we establish the smallest set of inequations $dc(In)$ containing In as well as $\{p \leq p : p \in Rec_{QS}(X)\}$ (reflexivity), and closed for transitivity, substitution, instantiation, recursion, and ω-induction (a well known infinitary axiom-scheme). This set is called the derivation-closure

Table 1
Inference rules

Substitution	$\frac{p_1 \leq p'_1, \ldots, p_k \leq p'_k}{f(p_1, \ldots, p_k) \leq f(p'_1, \ldots, p'_k)}$	for every $f \in \Sigma^k_{QS}$
Instantiation	$\frac{p \leq q}{p\rho \leq q\rho}$	for every substitution ρ
Recursion	$\frac{}{\mu x.p = p[\mu x.p/x]}$	
ω-Induction	$\frac{\text{for every } \pi \in Appr(p),\ \pi \leq q}{p \leq q}$	

of *In* and contains all the theorems we can derive from *In*. Thus, we write $\vdash p \leq q$ iff $p \leq q \in dc(In)$. It is worthwhile to recall the substitution, instantiation, recursion, and ω – induction rules (see Table 1).

Within this inequational inference system it is possible to use equations as abbreviations of the corresponding inequations (in both directions). That is, $p = q$ is an abbreviation of $p \leq q$ and $q \leq p$.

The idea is to identify a set $\mathcal{QS}$ of proper axioms such that $\vdash_{QS} p \leq q$ iff $\models p \leq q$, for all terms p and q in $cProc_{QS}(X)$. It is not difficult to arrive at the proper axioms in Table 2, where we assume that $a \neq b$, whenever a and b appears in the same axiom-scheme.

Several theorems can be derived from the proper axioms of Table 2, such as the distributivity of passive prefixing with respect to choice and parallel composition. Moreover, parallel composition was also explained and reflects the expected result that we cannot shuffle a passive action of one component between contiguous active actions of the other component without activating it.

Let $\mathcal{QS}^\dagger$ denote the subset of $\mathcal{QS}$ not containing the parallelism equations. It is easy to establish that $\mathcal{QS}$ is reductive over $\mathcal{QS}^\dagger$.

Proposition 16. *For every $p \in cProc_{QS}(X)$, there exists $\pi \in cProc^\dagger_{QS}(X)$ such that $\vdash_{QS} p = \pi$.*

Proof. We use induction on the structure of p for the finite terms. The proof for the recursive terms follows from ω – induction rule (see [7]). □

2.4. The automata semantics of QS processes

This section introduces the automata semantics for the QS model to complete the trinity of semantic models, together with the denotational and the axiomatic semantic domains already introduced in [7] and briefly presented in the previous sections. The automata semantics embodies the concepts of final and non-final states through the distinction between terms that denote processes in a quiescent state (final state) and terms that denote processes in a non-quiescent state (non-final state). There is an automaton for each term in $p \in cProc'_{QS}(X)$. The corresponding process (behaviour) is captured by the language recognized by the automaton. We also define a partial order relation between terms to prove full abstraction for $p \in cProc'_{QS}(X)$. The final result presented is the trinity in the sense of Hennessy, *i.e.*, equivalences among the three semantic domains (see [11]).

Recall that a (deterministic) finite automaton is defined as a quintuple $\mathcal{A} = \langle Q, q_0, \Sigma, \longrightarrow, F\rangle$, where Q is the finite set of states, $q_0 \in Q$ is the initial state, Σ is the alphabet, $\longrightarrow: Q \times \Sigma \to Q$ is the transition function, and $F \subseteq Q$ is the set of final or accepting states. Each element $\longrightarrow (p, a) = q$ is usually represented by $p \xrightarrow{a} q$. The graphical representation of an automaton is a labeled directed graph, where each node corresponds to a state – a term – and each arc corresponds to a transition. The initial state is represented by an extra incoming arrow and each final state is represented by a double circle. The relation $\longrightarrow_*$ is the reflexive and transitive closure of the relation $\longrightarrow$. As usual, the behaviour of an automaton $\mathcal{A} = \langle Q, q_0, \Sigma, \longrightarrow, F\rangle$ is described by its language, denoted by $L(\mathcal{A})$, and defined by $\{s \in \Sigma^* : \exists q \in F\ (q_0 \xrightarrow{s}_* q)\}$. Let *Aut* be the set of automata.

In order to formalize the automata semantics, first we define inductively a ternary relation $\longrightarrow$ using a collection of inference rules in the SOS style for each operator , which means that each rule has the form $\frac{premisses}{conclusion}$ [side condition]. The relation $\longrightarrow$ will be useful to obtain the set of states for each automaton.

Definition 17. The transition relation $\longrightarrow \subseteq cProc'_{QS}(X) \times Act \times cProc'_{QS}(X)$, is the least relation which satisfies the rules in Table 3.

Intuitively, $p \xrightarrow{a} q$ holds if the process p engages in the action a and then behaves like the process q. For each alphabet U, the process terms $abort_U$ and $stop_U$ have no rules, because the respective processes cannot engage in any action. The prefixing

Table 2
$\mathcal{QS}$ **Axiomatization**

Structure			
$x_U + x_U$	=	x_U	(S_{01})
$x_U + y_U$	=	$y_U + x_U$	(S_{02})
$x_U + (y_U + z_U)$	=	$(x_U + y_U) + z_U$	(S_{03})
$x_U + abort_U$	=	x_U	(S_{04})
$!a.(x_U + y_U)$	=	$!a.x_U + !a.y_U$	(S_{05})
Lattice			
$abort_U$	$\leq$	x_U	(Lat)
Rollback			
$!a.abort_U$	=	$abort_U$	(Rol)
Liveness			
$!a.x_U + stop_U$	=	$a.x_U$	(Liv)
Parallelism			
$x_U \| x_U$	=	x_U	(P_{01})
$x_U \| y_V$	=	$y_V \| x_U$	(P_{02})
$x_U \| (y_V \| z_W)$	=	$(x_U \| y_V) \| z_W$	(P_{03})
$x_U \| abort_V$	=	$abort_{U\cup V}$	(P_{04})
$stop_U \| stop_V$	=	$stop_{U\cup V}$	(P_{05})
$x_U \| (y_V + z_V)$	=	$x_U \| y_V + x_U \| z_V$	(P_{06})
$!a.x_U \| stop_{V\setminus\{a\}}$	=	$!a.(x_U \| stop_{V\setminus\{a\}})$	(P_{07})
$!a.x_{U\cup\{a\}} \| stop_{V\cup\{a\}}$	=	$abort_{U\cup V}$	(P_{08})
$!a.x_{U\setminus\{b\}} \| !b.y_{V\setminus\{a\}}$	=	$!a.(x_{U\setminus\{b\}} \| !b.y_{V\setminus\{a\}}) + !b.(!a.x_{U\setminus\{b\}} \| y_{V\setminus\{a\}})$	(P_{09})
$!a.x_{U\cup\{b\}} \| !b.y_{V\setminus\{a\}}$	=	$!a.(x_{U\cup\{b\}} \| !b.y_{V\setminus\{a\}})$	(P_{10})
$!a.x_{U\cup\{b\}} \| !b.y_{V\cup\{a\}}$	=	$abort_{U\cup V\cup\{a,b\}}$	(P_{11})
$!a.x_U \| !a.y_V$	=	$!a.(x_U \| y_V)$	(P_{12})

operators contain one rule (R1) with no premisses for each type of prefixing. The rules for choice (R2) are standard. The rules for the parallel operator (R3) are also standard, in the sense that there is a rule for the synchronization, whenever the first action for both processes is the same, and two symmetric rules for the interleaving, whenever the first action of each process is not in the alphabet of the other process. For recursion we have the standard unfolding rule (R4). It is easy to establish the following lemma.

Lemma 18. *For every* $p, q \in cProc'_{QS}(X)$ *and for every* $a \in Act$, $p \xrightarrow{a} q$ *implies* $\alpha(p) = \alpha(q)$.

Proof. By induction on the structure of $p \xrightarrow{a} q$ and considering Definition 8. □

We define the computation relation as the reflexive and transitive closure of $\longrightarrow$. Since we want that each state of an automaton correspond to a process term, and that the states be classified as final or non-final, we need to classify each process term according to its quiescence, mapping passive process terms into final states and active process terms into non-final states. Intuitively, an active prefixing term corresponds to a non-final state, since the process must perform some action, whereas a passive prefixing term corresponds to a final state, since the process may not engage in any action. Accordingly to this correspondence, we are going to define two sets of terms: the set of active terms, denoted by ◯, and the set of passive terms, denoted by ◎. Obviously, for each alphabet U, the term $abort_U$ corresponds to a non-final state and the term $stop_U$ corresponds to a final state. The choice corresponds to a final state, whenever at least one of its terms also correspond to a final state. The parallel composition corresponds to a final state, whenever both components also corresponds to final states. Formally, the set of process terms $Proc_{QS}(X)$ is partitioned into two subsets, ◯ and ◎, as follows.

Definition 19. The set of active terms ◯ and the set of passive terms ◎ are defined (simultaneously) as follows:

1. for all $U \in \mathcal{P}_{ne}(Act)$, $x_U, stop_U \in$ ◎ and $x_U, abort_U \in$ ◯;
2. $a.p \in$ ◎ and $!a.p \in$ ◯;
3. $p \in$ ◎ or $q \in$ ◎ implies $p + q \in$ ◎;
4. $p \in$ ◯ and $q \in$ ◯ implies $p + q \in$ ◯;
5. $p \in$ ◎ and $q \in$ ◎ implies $p \| q \in$ ◎;

Table 3
Transition rules for the relation $\longrightarrow$

R1. Prefixing

(1) $\dfrac{}{!a.p \xrightarrow{a} p}$ (2) $\dfrac{}{a.p \xrightarrow{a} p}$

R2. Choice

(1) $\dfrac{p \xrightarrow{a} p'}{p+q \xrightarrow{a} p'}$ (2) $\dfrac{q \xrightarrow{a} q'}{p+q \xrightarrow{a} q'}$

R3. Parallelism

(1) $\dfrac{p \xrightarrow{a} p' \qquad q \xrightarrow{a} q'}{p\|q \xrightarrow{a} p'\|q'}$

(2) $\dfrac{p \xrightarrow{a} p'}{p\|q \xrightarrow{a} p'\|q}\ [a \notin \alpha(q)]$ (3) $\dfrac{q \xrightarrow{a} q'}{p\|q \xrightarrow{a} p\|q'}\ [a \notin \alpha(p)]$

R4. Recursion

$$\frac{p[\mu x.p/x] \xrightarrow{a} q}{\mu x.p \xrightarrow{a} q}$$

6. $p \in \bigcirc$ or $q \in \bigcirc$ implies $p\|q \in \bigcirc$;
7. $p \in \circledcirc$ implies $\mu x.p \in \circledcirc$;
8. $p \in \bigcirc$ implies $\mu x.p \in \bigcirc$.

As a consequence of the above definition, given the alphabet $U = \{a, b\}$, the term $!a.b.stop_U \| \mu x_U.!a.b.x_U$ is in $\bigcirc$, whereas the terms $!a.b.stop_U + b.stop_U, \mu x_U.b.x_U$ are in $\circledcirc$. The proof of decidability for the predicates "$p \in \circledcirc$" or "$p \in \bigcirc$" is straightforward.

Now we are able to define the automata semantics for the terms in $cProc'_{QS}(X)$ as follows.

Definition 20. The automata semantics for process terms in $cProc'_{QS}(X)$ is a mapping

$$\mathcal{A}[\,] : cProc'_{QS}(X) \to Aut$$

which assigns to every $p \in cProc'_{QS}(X)$ the automaton

$$\mathcal{A}[p] = \langle S_p, p, \alpha(p), \longrightarrow\downarrow S_p, F_p \rangle,$$

where

$$S_p = \{q \in cProc'_{QS}(X) : \exists s \in \alpha(p)^* \ (p \xrightarrow{s}_* q)\},$$
$$\longrightarrow\downarrow S_p = \{q \xrightarrow{a} q' \in \longrightarrow : q, q' \in S_p \text{ and } a \in \alpha(p)\}$$
$$F_p = S_p \cap \circledcirc.$$

To construct the automaton for a given process term p, $\mathcal{A}[p]$, first we need to obtain the set of states that corresponds to the set of terms reachable by $\longrightarrow_*$ from p. The initial state is p. The final states are the set of states that correspond to passive terms. The set of transitions is obtained by restricting the transition relation $\longrightarrow$ to the set of states and to the alphabet of p.

We now introduce some examples. Let $U = \{coin, coffee, tea\}$.

Example 21. Consider the process term $machine_2$ given by

$$\mu x_U.coin.(!coffee.x_U + !tea.x_U)$$

that denotes a simple vending-machine which repeatedly serves either a *coffee* or a *tea* after receiving a *coin*. The corresponding automaton is given in Fig. 1. It has two states: 0 and 1 denoting, respectively, $machine_2$ and $!coffee.machine_2 + !tea.machine_2$. Notice that the choice between the active prefixing on *coffee* and *tea* actions means that one of them must occur, while the passive prefixing on *coin* action, means that it may occur. The process can be represented by the language recognized by the corresponding automaton, that is the language described by the regular expression $(coin\,(coffee + tea))^*$.

Example 22. Consider the process term given by

$$customer = !coin.coffee.stop_U,$$

which denotes the behaviour of a customer in the presence of the vending-machine from Example 21. The customer wants a *coffee* after inserting a *coin*. The corresponding automaton is given in Fig. 2. It has three states: *customer*, $coffee.stop_U$ and $stop_U$, denoted by 0, 1 and 2, respectively. Notice that the active prefixing on *coin* action means that the customer must insert

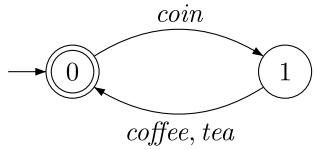

Fig. 1. Automaton associated to $machine_2$.

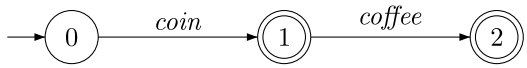

Fig. 2. Automaton associated to *customer*.

the *coin*, while the passive prefixing on *coffee* action, means that after inserting the *coin* it can deadlock when there is no *coffee*. The language recognized by the automaton is {*coin*, *coin coffee*}.

Example 23. Consider now the interaction of processes denoted by the terms $machine_2$ and *customer*. We assume that the customer gets *coffee* after having inserted a *coin*, and the machine is ready to serve it. The corresponding automaton is given in Fig. 3. It has three states 0, 1, and 2 denoting, respectively, the terms $machine_2 \| customer$, $(!coffee.machine_2 + !tea.machine_2) \| coffee.stop_U$, and $machine_2 \| stop_U$. Since the customer only wants coffee and the processes must synchronize in all actions, the *tea* action never occurs. Notice that 2 is the only final state because it corresponds to a parallel composition of passive terms.

Notice that the transition relation $\longrightarrow$ is deterministic. The seeming nondeterminism that can be derived by the choice rules (R2), never exists for process terms in $cProc'_{QS}(X)$. The respective restriction of process terms imposes that either one of the choice rules may be applied in each computation step.

However, this restricted set of process terms does not compromise the liveness properties described by the model. It is straightforward to prove that, for each term in $cProc_{QS}(X)$, there is a term in $cProc'_{QS}(X)$ equivalent to it, from the point of view of denotational or axiomatic semantics. The proof is done by structural induction: whenever there is a choice $p + q$, such that $a \in \nu(p) \cap \nu(q)$, we substitute $p + q$ by

(i) $p' + !a.(p'' + q'') + q'$ or
(ii) $p' + a.(p'' + q'') + q'$

where p' and q' are, respectively, the residuals of p and q, and p'' and q'' are the suffixes of a in p and q, respectively. Then we apply the induction hypothesis to $p'' + q''$. We use (i) whenever both p'' and q'' are prefixed with $!a.$, otherwise we use (ii).

In this way we can define the operational semantics for $cProc_{QS}(X)$ in the same way, allowing for non-determinism. But that would be equivalent, because for each non-deterministic automaton we could associate a canonical deterministic automaton which recognizes exactly the same language corresponding to the process term in $cProc'_{QS}(X)$ equivalent to the given process term in $cProc_{QS}(X)$.

Let $|Aut_{QS}| = \{\mathcal{A}[p] : p \in cProc'_{QS}(X)\}$. We now introduce some properties for the languages recognized by the automata in $|Aut_{QS}|$. But first, it is useful to state the following result:

Proposition 24. *For every $p \in cProc'_{QS}(X)$, for every $a \in Act$, and for every $s \in Act^*$,*

$$\frac{p \xrightarrow{as}_* q \qquad p \xrightarrow{a} r}{r \xrightarrow{s}_* q}.$$

Proof. Suppose that $p \xrightarrow{as}_* q$. According to $\longrightarrow_*$, there exists $r_1, \ldots, r_{|s|-1} \in cProc'_{QS}(X)$, such that $p \xrightarrow{a} r_1$, $r_1 \xrightarrow{s[1]} r_2, \ldots, r_{|s|-1} \xrightarrow{s[|s|]} q$, where $|s|$ denotes the length of s and $s[i]$ denotes the ith symbol of s, with $1 \le i \le |s|$. Since for every $p \in cProc'_{QS}(X)$, there is no distinct $q, r \in cProc'_{QS}(X)$ such that, $p \xrightarrow{a} q$ and $p \xrightarrow{a} r$, r must be r_1, and hence $r \xrightarrow{s}_* q$.

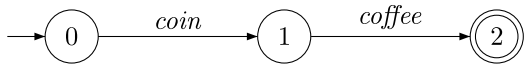

Fig. 3. Automaton associated to $machine_2 \| customer$.

As usual, ϵ denotes the empty word. It is straightforward to see that for a given alphabet U, $L(\mathcal{A}[abort_U]) = \emptyset$ and $L(\mathcal{A}[stop_U]) = \{\epsilon\}$.

Proposition 25. *For every $p, q \in cProc'_{QS}(X)$ and for every $a \in \alpha(p)$:*
1. $L(\mathcal{A}[!a.p]) = a.L(\mathcal{A}[p])$;
2. $L(\mathcal{A}[a.p]) = a.L(\mathcal{A}[p]) \cup \{\epsilon\}$;
3. $L(\mathcal{A}[p + q]) = L(\mathcal{A}[p]) \cup L(\mathcal{A}[q])$;
4. $L(\mathcal{A}[p \| q]) = L(\mathcal{A}[p])^{\alpha(p)} \| ^{\alpha(q)} L(\mathcal{A}[q])$.

Proof. See Appendix. □

We finish the subsection presenting a preorder relation over $|Aut_{QS}|$. The previous proposition is useful to prove that this relation is a Σ_{QS} – preorder over $|Aut_{QS}|$.

Definition 26. $\leq_{QS}$ is the relation over $|Aut_{QS}|$ which satisfies, for every $p, q \in cProc'_{QS}(X)$ the following conditions: (1) $\alpha(\mathcal{A}[p]) = \alpha(\mathcal{A}[q])$ and (2) $L(\mathcal{A}[p]) \subseteq L(\mathcal{A}[q])$.

Proposition 27. *$\leq_{QS}$ is a Σ_{QS} – preorder over $|Aut_{QS}|$.*

Proof. The monotonicity of the operators are easily obtained from the Proposition 25. □

We write $\Vdash_{QS} p \leq q$, whenever $\mathcal{A}[p] \leq_{QS} \mathcal{A}[q]$. We also write $\Vdash_{QS} p = q$, whenever $\Vdash_{QS} p \leq q$ and $\Vdash_{QS} q \leq p$.

2.5. The trinity

We now want to prove that the three previous semantic domains (denotational, axiomatic, and automata) are equivalent. The equivalence between the denotational and axiomatic semantic domains was already proved in [7] Here, we are going to state the equivalence between the automata and denotational semantic domains, by defining a semantics for the languages recognized by the automata in $|Aut_{QS}|$. Then we get the trinity after proving the equivalence between this language semantics and both the automata semantics and the denotational semantics.

The soundness of the proof system with respect to any interpretation in $\mathbb{QS}$ algebra which satisfies the appropriate inference rules is easy established.

Proposition 28. *For every $p, q \in Proc_{QS}(X)$, $\vdash_{QS} p \leq q$ implies $\models_{QS} p \leq q$.*

Proof. By induction on the length of derivation of theorem $\vdash_{QS} p \leq q$, after proving the soundness of the system $dc(\mathcal{QS})$.

Associativity of choice allows us to omit many brackets from terms without ambiguity. For instance, we write $p_1 + \cdots + p_k$ instead of $(\cdots(p_1 + p_2)\cdots + p_k)$. Commutativity and idempotence allow us to introduce some further notational convenience. If $\Theta = \{p_1, \ldots, p_k\}$ is a finite set of terms with alphabet U, the metaterm $+^U_{p\in\Theta}$ denotes the term $p_1 + \cdots + p_k$. If Θ is empty, $+^U_{p\in\Theta}$ simply denotes the term $abort_U$.

To prove the completeness of the inference system with respect to QS, restricted to $Proc_{QS}(X)$, as usual, first, we prove that the set of closed finite terms are reducible to normal forms. The suitable set of normal forms is defined as follows.

Definition 29. The set of normal forms of alphabet U, denoted by $NProc_{QS^U}(X)$, is inductively defined as follows:
1. $stop_U$ is a normal form of alphabet U;
2. given $C \subseteq U$, $C \neq \emptyset$, if for every $a \in C$, $n(a)$ is a normal form of alphabet U, then $+^U_{a\in C} a.n(a)$ and $+^U_{a\in C} !a.n(a)$ are normal forms of alphabet U.

Proposition 30. *For every $p \in cfProc^{\dagger}_{QS}(X)$, either $\vdash_{QS} p = abort_{\alpha(p)}$ or there exists a normal form $n(p)$ such that $\vdash_{QS} p = n(p)$.*

Proof. By induction on the structure of p. □

For the completeness proof, first, we prove the completeness for the set of normal forms, and later to finite terms.

Proposition 31. *Given the alphabet U, for every $\pi_1, \pi_2 \in NProc_{QS^U}(X)$, $\models_{QS} \pi_1 \leq \pi_2$ implies $\vdash_{fQS} \pi_1 \leq \pi_2$.*

Proof. By induction on the structure of π_1 and π_2. □

It is straightforward the proof for finite terms:

Proposition 32. *For every* $p, q \in cfProc_{QS}(X)$, $\models_{QS} p \leq q$ *implies* $\vdash_{QS} p \leq q$.

Proof. From Proposition 30, either $\vdash_{QS} p = abort_U$ or there exists $n(p) \in NProc_{QS^U}(X)$, such that $\vdash_{QS} p = n(p)$ or $\vdash_{QS} q = abort_U$, or there exists $n(q) \in NProc_{QS^U}(X)$, such that $\vdash_{QS} q = n(q)$. The proof follows from Propositions 28 and 31. □

Also straightforward is the completeness proof for the terms in $cProc_{QS}(X)$.

Proposition 33. *For every* $p, q \in cProc_{QS}(X)$, $\models_{QS} p \leq q$ *implies* $\vdash_{\mathbb{QS}} p \leq q$.

Proof. Suppose that $\models_{QS} p \leq q$. Let $\pi \in App(p)$. It is easy to establish that $\vdash_{QS} \pi \leq p$. By Proposition 28, we have $\models_{QS} \pi \leq p$, and by transitivity, $\models_{QS} \pi \leq q$. From finite approximation theorem [11], and by hypothesis, $\models_{QS} q^0 \leq \pi$. It is possible to show that, for every $n \geq 0$, $\vdash_{\mathbb{QS}} q^n \leq q^{n+1}$. By Proposition 28, we have $\models_{QS} q^0 \leq q^1 \leq q^2 \leq \cdots$. Thus, we conclude that there exists $\chi \in App(q)$, such that $\models_{QS} \pi \leq \chi$. From Proposition 28 and by ω-induction, we conclude $\vdash_{QS} p \leq q$.

Proposition 34. *For every* $p, q \in cProc_{QS}(X)$, $\vdash_{QS} p \leq q$ *iff* $\models_{\mathbb{QS}} p \leq q$.

Proof. Follows from Propositions 28 and 33. □

The algebra of languages recognized by the automata in $|Aut_{QS}|$ is defined as follows.

Definition 35. The algebra of languages recognized by the automata in $|Aut_{QS}|$, denoted by $\mathcal{L}_{QS}$, is the $\mathbb{QS}$ algebra, where for every $\langle U, L\rangle \in |\mathcal{L}_{QS}|$, U is an automaton alphabet and L is the corresponding language.

It is easy to establish the corresponding language semantics for the automata in $|Aut_{QS}|$ $\mathcal{L}_{QS}[\![-]\!] : Proc_{QS} \to (ENV_{\mathcal{L}_{QS}} \to |\mathcal{L}_{QS}|)$.

As usual, we write $\models^{\mathcal{L}}_{QS} p \leq q$ to denote $\mathcal{L}_{QS}[\![p]\!] \leq_{\mathcal{L}_{QS}} \mathcal{L}_{QS}[\![q]\!]$. We also write $\models^{\mathcal{L}}_{QS} p = q$ instead of $\models^{\mathcal{L}}_{QS} p \leq q$ and $\models^{\mathcal{L}}_{QS} q \leq p$.

It is straightforward to obtain the equivalence between the operational and the language semantics.

Proposition 36. *For every* $p, q \in cProc'_{QS}(X)$,

$$\models^{\mathcal{L}}_{QS} p \leq q \ \textit{iff} \ \Vdash_{QS} p \leq q.$$

Proof. It follows from the corresponding definitions of $\leq_{\mathcal{L}_{QS}}$ and $\leq_{QS}$. □

Also straightforward is the equivalence proof between the operational and the denotational semantics.

Proposition 37. *For every* $p, q \in cProc'_{QS}(X)$,

$$\Vdash_{QS} p \leq q \ \textit{iff} \ \models_{\mathbb{QS}} p \leq q.$$

Proof. It follows from the Proposition 36 and from the fact that $\models_{\mathcal{L}_{QS}} p \leq q$ iff $\models_{\mathbb{QS}} p \leq q$. □

From the previous Propositions 34 and 37 it is easy to obtain the trinity theorem.

Theorem 38. *For every* $p, q \in cProc'_{QS}(X)$, $\vdash_{QS} p \leq q$ *iff* $\models_{QS} p \leq q$ *iff* $\Vdash_{QS} p \leq q$.

3. The $QS_{\mathbf{box}}$ model

In this section we extend the Quiescence Model introduced in the previous section with an operator $[-]$ denoting a global liveness requirement. The operator $[-]$ applied to a process term induces a transactional behaviour until the execution of some determined action. The signature Σ_{QS} is enriched with this new global operator. The suitable denotational, axiomatic, and operational semantic domains are obtained through the introduction of new sets of traces, new axioms, and new transition rules, respectively. The final result is a trinity of equivalent semantics for the language in the sense of Hennessy. We also provide a conservative extension result for the previous axiomatic semantics. In this context, by being conservative, we mean that we cannot prove any other theorem about liveness in QS_{box} model which is not already provable in the QS model. The section is organized as the previous, that is, we first present the syntax of the process language and later the suitable

denotational, axiomatic, and automata semantics. It also ends with the equivalences among the the three semantic domains. Since the QS model is reductive over the restriction that excludes parallelism (see Proposition 16), we will not consider the parallelism in the new QS_{box} model.

3.1. The syntax of QS_{box} processes

Given a finite alphabet *Act* of actions, we introduce the signature for the QS_{box} model, denoted by $\Sigma_{QS_{\text{box}}}$, as $\Sigma^{\dagger}_{QS}$ (Definition 1) enriched with the set $\{[a] : a \in Act\}$. Intuitively, given a process term p and an action a, $[a]p$ denotes a transactional behaviour of p until the execution of a. We assume that this new operator binds stronger than the others.

The set of recursive terms over QS_{box}, denoted by $Rec_{QS_{\text{box}}}(X)$, is defined as in the previous section for the signature Σ_{QS} (see Definition 2). We define subterms *mutatis mutandis* as we did before (see Definition 4). Similarly, Definitions 3 and 5 can be extended by defining, respectively, $\alpha([a]p) = \{a\} \cup \alpha(p)$ and $\phi([a]p) = \phi(p)$. The language of processes with global liveness requirements is defined as follows.

Definition 39. The set of processes terms with global liveness requirements, denoted by $Proc_{QS_{\text{box}}}(X)$, is the subset of $Rec_{QS_{\text{box}}}(X)$, where each term p satisfies the following conditions:

1. is guarded;
2. every subterm $!a.q$, $a.q$ or $[a]p$ of p satisfies $a \in \alpha(q)$;
3. every subterm $q + r$ of p satisfies $\alpha(q) = \alpha(r)$;
4. every subterm $\mu x.q$ of p satisfies $\alpha(x) = \alpha(q)$.

Notice that, as a consequence of the last definition, the (non-passively) guarded terms $\mu x.!tick.x$ and $[tick]\mu x.!tick.x$ are in $Proc_{QS_{\text{box}}}(X)$, assuming that $tick \in \alpha(x)$, whereas the term $[tick]\mu x.x$ is not in $Proc_{QS_{\text{box}}}(X)$, because it is not guarded.

Let $Proc^{\ddagger}_{QS_{\text{box}}}(X)$ denotes the subset of process terms in $Proc_{QS_{\text{box}}}(X)$ and also in $Rec^{\dagger}_{QS}(X)$. As usual, whenever necessary we also use the prefix notation c and f. As we did in the previous section, we also define a subset of process terms in $cProc_{QS_{\text{box}}}(X)$, denoted by, $cProc'_{QS_{\text{box}}}(X)$ (see Definition 10).

3.2. The denotational semantics of QS_{box} processes

We are going to define a denotational trace semantics for the QS_{box} model as an extension of the corresponding denotational semantics for the QS model adapted to the terms in $Proc_{QS_{\text{box}}}(X)$. We add a suitable interpretation for the new global liveness operator based on the forthcoming definition. We still continue having an algebraic cpo for the processes.

Definition 40. Let U be an alphabet, $L \subseteq U^*$ and $a \in U$. The set of words of L containing at least one occurrence of a, denoted by $L_{[a]}$, is $\{s \in L : \exists s', s'' \in U^* \ (s = s'as'')\}$.

It is easy to establish the following properties for the previous set operator.

Proposition 41. *Let U and V be alphabets, L and L' subsets of U^*, T a subset of V^*, and $a, b \in U \cup V$, such that $a \neq b$. We have that*:

1. $\emptyset_{[a]} = \emptyset$;
2. $\{a\}_{[a]} = \{a\}$; $\{b\}_{[a]} = \emptyset$;
3. $(a.L)_{[a]} = a.L$;
4. $(b.L)_{[a]} = b.L_{[a]}$;
5. $(L \cup L')_{[a]} = L_{[a]} \cup L'_{[a]}$.

Proof. See Appendix. □

The envisaged $\Sigma_{QS_{\text{box}}}$ – domain is established as follows.

Definition 42. The algebra of process terms with global liveness requirements is the triple

$$\mathbb{QS}_{\text{box}} = \langle |\mathbb{QS}_{\text{box}}|, \leq_{QS_{\text{box}}}, \Sigma_{\mathbb{QS}_{\text{box}}} \rangle,$$

where

- $|\mathbb{QS}_{\text{box}}| = \{\langle U, L\rangle : U \in \mathcal{P}_{ne}(Act) \text{ and } L \subseteq U^*\}$
- $\langle U, L\rangle \leq_{\mathbb{QS}_{\text{box}}} \langle U', L'\rangle$ iff $U = U'$ and $L \subseteq L'$
- $\Sigma_{\mathbb{QS}_{\text{box}}} = \{abort^{U}_{\mathbb{QS}_{\text{box}}}, stop^{U}_{\mathbb{QS}_{\text{box}}} : U \in \mathcal{P}_{ne}(Act)\} \cup \{!a._{\mathbb{QS}_{\text{box}}}, a._{\mathbb{QS}_{\text{box}}}, [a]_{\mathbb{QS}_{\text{box}}} : a \in Act\} \cup \{+_{\mathbb{QS}_{\text{box}}}\}$, where

Table 4
$\mathcal{QS}_{\text{box}}$ **Axiomatization – Global liveness proper axioms**

$[a]abort_U$	=	$abort_U$	(GL_{01})
$[a]stop_U$	=	$abort_U$	(GL_{02})
$[a]!a.x_U$	=	$!a.x_U$	(GL_{03})
$[a]!b.x_U$	=	$!b.[a]x_U$	(GL_{04})
$[a](x_U + y_U)$	=	$[a]x_U + [a]y_U$	(GL_{05})

$$
\begin{array}{rl@{\qquad}rl}
abort^U_{\mathbb{QS}_{\text{box}}} & : \to |\mathbb{QS}| & abort^U_{\mathbb{QS}_{\text{box}}} & = \langle U, \emptyset\rangle \\
stop^U_{\mathbb{QS}_{\text{box}}} & : \to |\mathbb{QS}_{\text{box}}| & stop^U_{\mathbb{QS}_{\text{box}}} & = \langle U, \{\epsilon\}\rangle \\
!a._{\mathbb{QS}_{\text{box}}} & : |\mathbb{QS}_{\text{box}}| \to |\mathbb{QS}_{\text{box}}| & !a._{\mathbb{QS}_{\text{box}}}\langle U, L\rangle & = \langle \{a\} \cup U, a.L\rangle \\
a._{\mathbb{QS}_{\text{box}}} & : |\mathbb{QS}_{\text{box}}| \to |\mathbb{QS}_{\text{box}}| & a._{\mathbb{QS}_{\text{box}}}\langle U, L\rangle & = \langle \{a\} \cup U, \{\epsilon\} \cup a.L\rangle \\
[a]_{\mathbb{QS}_{\text{box}}} & : |\mathbb{QS}_{\text{box}}| \to |\mathbb{QS}_{\text{box}}| & [a]_{\mathbb{QS}_{\text{box}}}\langle U, L\rangle & = \langle \{a\} \cup U, L_{[a]}\rangle \\
+_{\mathbb{QS}_{\text{box}}} & : |\mathbb{QS}_{\text{box}}|^2 \to |\mathbb{QS}_{\text{box}}| & \langle U, L\rangle +_{\mathbb{QS}_{\text{box}}}\langle U', L'\rangle & = \langle U \cup U', L \cup L'\rangle
\end{array}
$$

It is straightforward to prove that $\langle |\mathbb{QS}_{\text{box}}|, \leq_{\mathbb{QS}_{\text{box}}}, \Sigma_{\mathbb{QS}_{\text{box}}}\rangle$ is a $\Sigma_{QS_{\text{box}}}$-domain, with $\langle |\mathbb{QS}_{\text{box}}|, \leq_{\mathbb{QS}_{\text{box}}}\rangle$ an algebraic cpo.

Example 43. Consider the process term $[coffee]machine_2$, where $machine_2$ is in Example 21, which denotes the recursive behaviour of a vending-machine that is able to serve a *coffee* or a *tea* after receiving a *coin* and also guarantees that at least one *coffee* is served.

$$
\begin{aligned}
\mathbb{QS}_{\text{box}}[\![[coffee]machine_2]\!] &=_{\mathbb{QS}_{box}} [coffee]_{\mathbb{QS}_{box}}\mathbb{QS}_{\text{box}}[\![machine_2]\!] \\
&=_{\mathbb{QS}_{box}} [coffee]_{\mathbb{QS}_{box}}\langle U, \{coin\,coffee,\ coin\,tea\}^*\rangle \\
&=_{\mathbb{QS}_{box}} \langle U, (\{coin\,coffee,\ coin\,tea\}^*)_{[coffee]}\rangle \\
&=_{\mathbb{QS}_{box}} \langle U, \{coin\,coffee\}^+\rangle
\end{aligned}
$$

Example 44. Consider now the process term $machine_3$ given by

$$\mu x_U.[coffee]coin.(!coffee.x_U + !tea.x_U),$$

which denotes a vending-machine (a variant of process denoted by $machine_2$) that is recursively compelled to provide *coffee* after receiving a *coin*. The $\mathbb{QS}_{\text{box}}$ semantics of p is given by the least fixed point of the equation $\xi = \mathbb{QS}_{\text{box}}[\![machine_3]\!][\xi/x_U]$. Accordingly to the *finite approximating theorem* (see [11]) and because all finite approximations of $machine_3$ are equivalent to $abort_U$, we conclude that $\mathbb{QS}_{\text{box}}[\![machine_3]\!] =_{\mathbb{QS}_{box}} \langle U, \emptyset\rangle$.

The process, recursively, after *coffee* or *tea* is eager to execute *coffee* but it needs to execute *coin* first. Notice that the process never reaches a quiescent state. Since in this model traces must be finite, so it is not possible to get the desired semantics for $machine_3$. Therefore, the suitable set of quiescent traces must be empty. In a different paper [10] we extend the results for infinite commitments, defining process denotational semantics with sets of possible infinite sequences, as in [4].

3.3. The axiomatic semantics of QS_{box} processes

We now give a syntactical characterization of the QS_{box} model through a suitable sound and complete inequational inference system. We use the inference system of QS model adapted to the terms in $Proc_{QS_{\text{box}}}(X)$, adding a new set of proper axioms suitable to global liveness and removing the parallelism proper axioms. The proper axioms for the global liveness are in Table 4, where as usual we assume that $a \neq b$, whenever a and b appear in the same axiom-scheme.

Several theorems can be derived from these proper axioms, such as $\vdash_{QS_{\text{box}}} [a]a.x_U = !a.x_U$ or $\vdash_{QS_{\text{box}}} [a]b.x_U = !b.[a]x_U$. It is also possible derive from $dc(QS_{\text{box}})$ other theorems such as $\vdash_{QS_{\text{box}}} [a][a]x_U = [a]x_U$ or $\vdash_{QS_{\text{box}}} [a][b]x_U = [b][a]x_U$, where $x_U \in cProc_{QS_{\text{box}}}(X)$. With these theorems we can reduce the successive [-] – actions of the same action.

Example 45. Let us consider the process term $machine_3$ from Example 44.

$$
\begin{aligned}
\vdash_{QS_{\text{box}}} \quad machine_3 &= [coffee]coin.(!coffee.machine_3 + !tea.machine_3) \\
&= !coin.[coffee](!coffee.machine_3 + !tea.machine_3) \\
&= !coin.!coffee.machine_3 + !coin.!tea.[coffee]machine_3)
\end{aligned}
$$

Since all finite approximations of are $abort_U$, by ω-induction it follows:

$$\vdash_{QS_{\text{box}}} machine_3 = abort_U.$$

Let $\mathcal{QS}^{\ddagger}_{\text{box}}$ denote the subset of $\mathcal{QS}_{\text{box}}$ not containing the global activity proper axioms. It is easy to establish that $\mathcal{QS}_{\text{box}}$ is reductive over $\mathcal{QS}^{\ddagger}_{\text{box}}$.

Proposition 46. *For every $p \in cProc_{QS_{\text{box}}}(X)$, there exists $\pi \in cProc^{\ddagger}_{QS_{\text{box}}}(X)$ such that $\vdash_{QS_{\text{box}}} p = \pi$.*

Proof. See Appendix. □

From the previous proposition it is also proved that $\mathcal{QS}_{\text{box}}$ is a conservative extension of $\mathcal{QS}^{\dagger}$.

Proposition 47. (1) *For every $p \in Proc_{QS_{\text{box}}}(X)$, there exists $q \in Proc_{QS}(X)$, such that $\vdash_{QS_{\text{box}}} p = q$.* (2) *For every $p, q \in Proc_{QS}X)$, $\vdash_{QS_{\text{box}}} p = q$ implies $\vdash_{QS} p = q$.*

3.4. The automata semantics of QS_{box} processes

We now want to give an automata semantics for the QS_{model} to complete Hennessy's trinity together with the denotational and axiomatic semantics already presented in the two previous subsections. As expected, we are going to adapt the QS automata semantics for process terms in $cProc'_{QS_{\text{box}}}(X)$. For the transition relation, we need two more rules for the new operator (see Table 5). The set of active terms $\bigcirc$ and the set of passive terms $\circledcirc$ are defined as we did before for the QS model (see Definition 19). Just a new case is added to the definition relative to the global liveness, $[a]p \in \bigcirc$.

However, when we tried to define the automata semantics to a process term in $cProc'_{QS_{\text{box}}}(X)$ non passively guarded, such as $machine_3$ from Example 44, we found a new aspect that did not happen in the QS model. Every time we use Rule R5-(2), a new process term prefixed with the new global operator is derived. When we have a recursive process, the rule can be used infinitely often, and it is possible to derive infinite terms. Thus, it is possible to obtain infinite sets of terms, *i.e.*, infinite sets of states for automata. To continue having finite automata, as for the QS model, we decided, first, to group the terms in congruence classes according to the following relation.

Definition 48. The (congruence) relation in $cProc'_{QS_{\text{box}}}(X)$, denoted by $\equiv$, is the least congruence$-\Sigma_{QS_{\text{box}}}$ relation induced by the following equations, for every $a, b \in Act$ and $p \in cProc'_{QS_{\text{box}}}(X)$: (1) $[a]p = [a][a]p$ and (2) $[a][b]p = [b][a]p$.

We denote each equivalence class either by its irreducible representative or by one permutation. From now on, in the context of this semantic domain, a term represents a class of equivalent terms, corresponding to a state of an automaton. The relation $\equiv$ is of finite index, and thus, every automaton is finite, *i.e.*, it has a finite set of states. The transition relation $\longrightarrow_{\text{box}}$ is defined as follows, where as usual, we assume that $a \neq b$, whenever a and b appear in the same inference rule.

Definition 49. The transition relation $\longrightarrow_{\text{box}} \subseteq (cProc'_{QS_{\text{box}}}(X)/\equiv) \times Act \times (cProc'_{QS_{\text{box}}}(X))/\equiv$, is the least relation which satisfies the rules in Table 3, adapted to the process terms in $cProc'_{QS_{\text{box}}}(X)$ and also the rules in Table 5.

Now we are able to define the automata semantics for the terms in $Proc'_{QS_{\text{box}}}(X)$ as follows.

Definition 50. The automata semantics for process terms in $cProc'_{QS_{\text{box}}}(X)$ is a mapping

$$\mathcal{A}_{\text{box}}[-] : (cProc'_{QS_{\text{box}}}(X)/\equiv) \to Aut$$

which assigns to every $p \in cProc'_{QS_{\text{box}}}(X)$ the automaton

$$\mathcal{A}_{\text{box}}[p] = \langle S_p, p, \alpha(p), \longrightarrow_{\text{box}} \downarrow S_p, F_p \rangle,$$

Table 5
Global liveness transition rules for the $\longrightarrow$ relation

R5. Global liveness	
(1) $\dfrac{p \xrightarrow{a}_{\text{box}} q}{[a]p \xrightarrow{a}_{\text{box}} q}$	(2) $\dfrac{p \xrightarrow{b}_{\text{box}} q}{[a]p \xrightarrow{b}_{\text{box}} [a]q}$

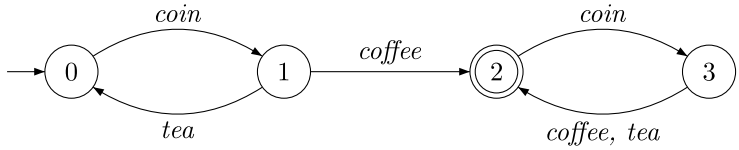

Fig. 4. Automaton associated to $[coffee]machine_2$.

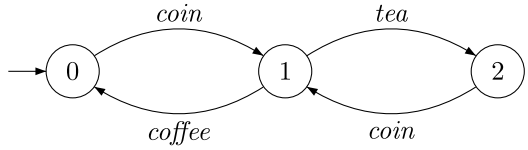

Fig. 5. Automaton associated to $machine_3$.

where

$$S_p = \{q \in cProc'_{QS_{box}}(X) : \exists s \in \alpha(p)^* \ (p \xrightarrow{s}{}^*_{box} q)\}/\equiv$$
$$\longrightarrow_{box} \downarrow S_p = \{q \xrightarrow{a}_{box} q' \in \longrightarrow_{box} : q, q' \in S_p \text{ and } a \in \alpha(p)\}$$
$$F_p = (\{q \in cProc'_{QS_{box}}(X) : \exists s \in \alpha(p)^* \ (p \xrightarrow{s}{}^*_{box} q)\} \cap \circledcirc)/\equiv$$

Let $|Aut_{QS_{box}}| = \{\mathcal{A}[p] : \ p \in cProc'_{QS_{box}}(X)\}$. The properties for the languages recognized by the automata in $|Aut_{QS}|$ are easily adapted to the automata in $|Aut_{QS_{box}}|$ (see Proposition 25). It is worthwhile to present the property concerning the global operator.

Proposition 51. *For every $p \in cProc'_{QS_{box}}(X)$ and for every $a \in \alpha(p)$, $L(\mathcal{A}[[a]p]) = (L(\mathcal{A}[p]))_{[a]}$.*

Proof. See Appendix. □

We define the relation $\leq_{QS_{box}}$ over $|Aut_{QS_{box}}|$ as we did for the automata in $|Aut_{QS}|$.

Definition 52. $\leq_{QS_{box}}$ is the relation over $|Aut_{QS_{box}}|$ which satisfies, for every $p, q \in cProc'_{QS_{box}}(X)$ the following conditions: (1) $\alpha(\mathcal{A}[p]) = \alpha(\mathcal{A}[q])$ and (2) $L(\mathcal{A}[p]) \subseteq L(\mathcal{A}[q])$.

Proposition 53. *$\leq_{QS_{box}}$ is a $\Sigma_{QS_{box}}$-preorder over $|Aut_{QS_{box}}|$.*

Proof. Since the global operator is monotonic, the proof is straightforward. □

We write $\Vdash_{QS_{box}} p \leq q$, whenever $\mathcal{A}[p] \leq_{QS_{box}} \mathcal{A}[q]$. We also write $\Vdash_{QS_{box}} p = q$, whenever $\Vdash_{QS_{box}} p \leq q$ and $\Vdash_{QS_{box}} q \leq p$.

Example 54. Consider the process term $[coffee]machine_2$ from Example 43. The associated automaton is given in Fig. 4, where 0, 1, 2 and 3 denote, respectively, $[coffee]\ machine_2$, $[coffee](!coffee.machine_2 + !tea.machine_2)$, $machine_2$ and $!coffee.machine_2$. To guarantee that the machine serves at least a *coffee* after receiving a *coin*, we need two more states then for the automaton associated to the process term $machine_2$. Thus the automaton has plus non-final states until to be guaranteed that the *coffee* is performed. The corresponding recognized language is the regular language $(coin\ tea)^*\ coin\ coffee(coin\ coffee + coin\ tea))^*$.

Example 55. Consider now $machine_3$ from Example 44. The associated automaton is given in Fig. 5, where 0, 1 and 2 denote respectively, $machine_3$, $[coffee](!coffee.machine_3 + !tea.machine_3)$ and $[coffee]machine_3$. The recognized language is $\emptyset$.

3.5. The trinity

We now want to prove the equivalences among the denotational, axiomatic, and automata semantic domains presented in the last three sections. First, we prove the equivalence between the denotational and the axiomatic domains based on the

corresponding equivalence proof for the QS model (see [7]). Then, we prove the equivalence between the denotational and the automata semantics.

The soundness of the proof system with respect to any interpretation in $\mathbb{QS}_{\text{box}}$ algebra which satisfies the appropriate inference rules is established as for QS model.

Proposition 56. *For every* $p, q \in Proc_{\mathcal{QS}_{\text{box}}}(X)$, $\vdash_{\mathcal{QS}_{\text{box}}} p \leq q$ *implies* $\models_{\mathcal{QS}_{\text{box}}} p \leq q$.

Proof. By induction on the length of derivation of theorem $\vdash_{\mathcal{QS}_{\text{box}}} p \leq q$, after proving the soundness of the system $dc(\mathcal{QS}_{\text{box}})$. Proposition 41 is used to prove the soundness of the global liveness proper axioms. □

Completeness is obtained as we did in the previous section for the QS model (see Proposition 33).

Proposition 57. *For every* $p, q \in cProc_{\mathcal{QS}_{\text{box}}}(X)$, $\models_{\mathcal{QS}_{\text{box}}} p \leq q$ *implies* $\vdash_{\mathbb{QS}_{\text{box}}} p \leq q$.

The equivalence of these semantic domains is straightforward.

Proposition 58. *For every* $p, q \in cProc_{\mathcal{QS}_{\text{box}}}(X)$, $\vdash_{\mathcal{QS}_{\text{box}}} p \leq q$ *iff* $\models_{\mathbb{QS}_{\text{box}}} p \leq q$.

Proof. Follows from Propositions 56 and 57. □

The equivalence between the operational and denotational semantics is also obtained as we did in the previous section for the QS model (see Proposition 37).

Proposition 59. *For every* $p, q \in cProc'_{\mathcal{QS}_{\text{box}}}(X)$,

$\Vdash_{\mathcal{QS}_{\text{box}}} p \leq q$ *iff* $\models_{\mathcal{QS}_{\text{box}}} p \leq q$.

Thus, we conclude the trinity for the $\mathcal{QS}_{\text{box}}$ model.

Theorem 60. *For every* $p, q \in cProc'_{\mathcal{QS}_{\text{box}}}(X)$,

$\vdash_{\mathcal{QS}_{\text{box}}} p \leq q$ *iff* $\models_{\mathcal{QS}_{\text{box}}} p \leq q$ *iff* $\Vdash_{\mathcal{QS}_{\text{box}}} p \leq q$.

4. Conclusions and further work

We provided an automata semantics for the language of passively guarded terms presented for the QS model [7]. For each process, there is a finite automaton that embodies the concepts of final and non-final states through the distinction between terms that denote processes in a quiescent (final) state and terms that denote processes in a non-quiescent (non-final) state. A process is captured by the language recognized by the respective automaton. A partial order relation between processes is defined in order to prove full abstraction. Thus, we completed Hennessy's trinity with an operational semantics, together with the denotational and the axiomatic semantics already defined in [7].

We also extended the QS model with the new operator $[-]$ denoting a global liveness requirement such that, when applied to a process p, it induces a transactional behaviour until the execution of some specified action. We defined the suitable denotational axiomatic, and automata semantic domains. The final result is another trinity of equivalent semantics for process terms. We also proved that this model in the axiomatic domain is a conservative extension of the axiomatic domain for the QS model. As a consequence, we conclude that the denotational and automata semantics were also conservative extensions of the corresponding domains for the QS model restricted to the terms syntactically written without the parallel operator.

The QS model needs to be reformulated to capture infinite liveness requirements. Some difficulties do arise when we try to capture within the proposed models, with infinite liveness requirements, such as the *clock* process described by the term $clock = \mu x_{\{tick\}}.!tick.x_{\{tick\}}$, that has the recursive commitment of ticking. For example, for the clock process, there is no trace in $\{tick\}^*$ after which the clock is in a quiescent state. In fact, the QS model imposes that the set of recursive process terms is a set of passively guarded recursive process terms such that the behaviour of each process is captured by a (finite or infinite) set of finite traces. Thus, it is not possible to obtain the suitable semantics for non-passively guarded process terms with these tools through that sets of finite traces. Actually the clock process must perform a transaction that could not be committed, meaning that the corresponding set of quiescent traces is empty. To capture infinite commitments in the denotational domain we need to consider infinite sequences such that, the behaviour of the clock process performs $\{tick\}^\omega$. We are now aware of a suitable operational semantics based on Büchi automata (see, *e.g.*, [32]) in order to have, as in the previous models, non-final states denoting the active process terms and final states denoting the passive process terms. The main goal is to construct three equivalent semantic domains (operational, axiomatic, and denotational) for a new language of process terms that captures infinite behaviours.

The idea of extending the process signature with a new global operator inspired in the Temporal Logic operator $\Diamond$, can be the starting point in establishing a bridge from Temporal Logic to a Process Algebra. New liveness properties inspired in temporal operators can then be added to the QS model or to the reformulated QS model with infinite liveness requirements. New global operators, such as $[a^*]$, for a given action a, inducing infinite transactional behaviour of performing a infinitely often, can be added to the process signature.

Acknowledgements

This work was partially supported by LabMAg and by FCT SFRH/BD/9046/2002. José Félix Costa was supported by FEDER and FCT Pluriannual 2007. The authors are grateful to Luís Caires for useful comments.

Appendix (Proofs)

Proof (*Proposition 25*).

1. Given $p \in cProc'_{QS}(X)$, such that $\mathcal{A}[p] = \langle S_p, p, \alpha(p), \longrightarrow\downarrow S_p, F_p\rangle$, we have that

   $\mathcal{A}[!a.p] = \langle S_{!a.p}, !a.p, \alpha(!a.p), \longrightarrow\downarrow S_{!a.p}, F_{!a.p}\rangle$,

   where:

   $$\begin{aligned} S_{!a.p} &= \{!a.p\} \cup S_p \\ \alpha(!a.p) &= \alpha(p) \\ \longrightarrow\downarrow S_{!a.p} &= \{!a.p \xrightarrow{a} p\} \cup \ \longrightarrow\downarrow S_p \\ F_{!a.p} &= F_p = S_p \cap \circledcirc \end{aligned}$$

   Suppose that $s \in L(\mathcal{A}[!a.p])$, *i.e.*, there exists $t \in F_{!a.p}$, such that $!a.p \xrightarrow{s}_* t$. Since $!a.p \in \bigcirc$, s can not be ϵ. It follows by R1–(1) that s is as', with $s' \in \alpha(p)^*$. So, $!a.p \xrightarrow{a} p$ and we conclude $p \xrightarrow{s'}_* t$. Therefore, $s' \in L(\mathcal{A}[p])$ and hence, $s \in a.L(\mathcal{A}[p])$.

   Conversely, suppose that s is as', with $s' \in L(\mathcal{A}[p])$. There exists $t \in F_p$, such that $p \xrightarrow{s'}_* t$. Since $F_p = F_{!a.p}$ we conclude by R1–(1) and the transitivity closure of $\longrightarrow$ that $!a.p \xrightarrow{as'}_* t$, *i.e.*, $s \in L(\mathcal{A}[!a.p])$.
2. Given $p \in cProc'_{QS}(X)$, such that $\mathcal{A}[p] = \langle S_p, p, \alpha(p), \longrightarrow\downarrow S_p, F_p\rangle$, we have that

   $\mathcal{A}[a.p] = \langle S_{a.p}, a.p, \alpha(a.p), \longrightarrow\downarrow S_{a.p}, F_{a.p}\rangle$,

   where:

   $$\begin{aligned} S_{a.p} &= \{a.p\} \cup S_p \\ \alpha(a.p) &= \alpha(p) \\ \longrightarrow\downarrow S_{a.p} &= \{a.p \xrightarrow{a} p\} \cup S_p \\ F_{a.p} &= \{a.p\} \cup F_p \end{aligned}$$

   Suppose that $s \in L(\mathcal{A}[a.p])$, *i.e.*, there exists $t \in F_{a.p}$, such that $a.p \xrightarrow{s} t$. If t is $a.p$, s is ϵ, otherwise, s is as', with $s' \in \alpha(p)^*$. Therefore, by R1–(2), $a.p \xrightarrow{a} p$, and assuming that $a.p \xrightarrow{s} t$, we conclude that $p \xrightarrow{s'} t$. Thus, $s' \in L(\mathcal{A}[p])$ and hence, $s \in a.L(\mathcal{A}[p])$.

   Conversely, suppose that $s \in a.L(\mathcal{A}[p]) \cup \{\epsilon\}$. If s is ϵ, then $a.p \xrightarrow{\epsilon} a.p$. Since $a.p \in F_{a.p}$, we conclude that $\epsilon \in L(\mathcal{A}[a.p])$. If s is as', with $s' \in L(\mathcal{A}[p])$, then there exists $t \in F_p$, such that $p \xrightarrow{s'} t$. Therefore, $a.p \xrightarrow{as'} t$, with $t \in F_{a.p}$, and hence, $s \in L(\mathcal{A}[a.p])$.
3. Given $p, q \in cProc'_{QS}(X)$, such that $\mathcal{A}[p] = \langle S_p, p, \alpha(p), \longrightarrow\downarrow S_p, F_p\rangle$ and $\mathcal{A}[q] = \langle S_q, q, \alpha(q), \longrightarrow\downarrow S_q, F_q\rangle$, we have that

   $\mathcal{A}[p+q] = \langle S_{p+q}, p+q, \alpha(p+q), \longrightarrow\downarrow S_{p+q}, F_{p+q}\rangle$

   where

   $$\begin{aligned} S_{p+q} &= \{p+q\} \cup S_p\backslash\{p\} \cup S_q\backslash\{q\} \\ \alpha(p+q) &= \alpha(p) \ = \ \alpha(q) \\ \longrightarrow\downarrow S_{p+q} &= \{p+q \xrightarrow{a} p' : p \xrightarrow{a} p' \in \longrightarrow\downarrow S_p\} \cup \\ &\quad \{p+q \xrightarrow{a} q' : p \xrightarrow{a} q' \in \longrightarrow\downarrow S_q\} \cup \\ &\quad \longrightarrow\downarrow S_p\backslash\{p \xrightarrow{a} p' : a \in \nu(p)\} \cup \\ &\quad \longrightarrow\downarrow S_q\backslash\{q \xrightarrow{a} q' : a \in \nu(q)\} \\ F_{p+q} &= (\{p+q\} \cap \circledcirc) \cup F_p\backslash\{p\} \cup F_q\backslash\{q\} \end{aligned}$$

   Suppose that $s \in L(\mathcal{A}[p+q])$, *i.e.*, there exists $t \in F_{p+q}$, such that $p+q \xrightarrow{s} t$. The proof is done by induction on the length of s. If s is ϵ, then t is $p+q$, and thus, $p \in \circledcirc$ or $q \in \circledcirc$. Therefore, $p \xrightarrow{\epsilon} p$ or $q \xrightarrow{\epsilon} q$, *i.e.*, $\epsilon \in L(\mathcal{A}[p]) \cup L(\mathcal{A}[q])$. If

s is as', then $p+q \xrightarrow{a} r$, assuming that there is $r \in S_{p+q}$, such that either $p \xrightarrow{a} r$ or $q \xrightarrow{a} r$. For both situations, when we apply the rule of Proposition 24, we conclude that $r \xrightarrow{s'} t$. Therefore, either $p \xrightarrow{as'} t$ or $q \xrightarrow{as'} t$. Thus, $s \in L(\mathcal{A}[p])$ or $s \in L(\mathcal{A}[q])$.
Conversely, suppose that $s \in L(\mathcal{A}[p]) \cup L(\mathcal{A}[q])$. The proof is also done by induction on the length of s. We have that either $s \in L(\mathcal{A}[p])$ or $s \in L(\mathcal{A}[q])$. We are just going to prove the first case, $s \in L(\mathcal{A}[p])$. The proof for $s \in L(\mathcal{A}[q])$ is similar. If s is ϵ, since $p \in F_p$ then we conclude $p+q \in F_{p+q}$, *i.e.*, $p+q \xrightarrow{\epsilon} p+q$. Hence, $\epsilon \in L(\mathcal{A}[p+q])$.
If s is as', then there exists $t \in F_p$, such that $p \xrightarrow{as'} t$. Assuming that $p \xrightarrow{a} p'$ we conclude that $p+q \xrightarrow{a} p'$, and consequently, $p+q \xrightarrow{as'} t$. Thus, $s \in L(\mathcal{A}[p+q])$.

4. Given $p, q \in cProc'_{QS}(X)$, such that $\mathcal{A}[p] = \langle S_p, p, \alpha(p), \longrightarrow\downarrow S_p, S_p \cap \circledcirc\rangle$ and $\mathcal{A}[q] = \langle S_q, q, \alpha(q), \longrightarrow\downarrow S_q, S_q \cap \circledcirc\rangle$, we have that

$$\mathcal{A}[p\|q] = \langle S_{p\|q}, p\|q, \alpha(p\|q), \longrightarrow\downarrow S_{p\|q}, S_{p\|q} \cap \circledcirc\rangle,$$

where:

$$\begin{aligned}
S_{p\|q} &\subseteq \{p'\|q' : p' \in S_p,\ q' \in S_q\}\\
\alpha(p\|q) &= \alpha(p) \cup \alpha(q)\\
\longrightarrow\downarrow S_{p\|q} &= \{p'\|q' \xrightarrow{a} p''\|q'' : p' \xrightarrow{a} p'' \in \longrightarrow\downarrow S_p,\ q' \xrightarrow{a} q'' \in \longrightarrow\downarrow S_q\} \cup\\
&\quad \cup \{p'\|q' \xrightarrow{a} p''\|q' : p' \xrightarrow{a} p' \in \longrightarrow\downarrow S_p,\ q' \xrightarrow{a} q'' \notin \longrightarrow\downarrow S_q\} \cup\\
&\quad \cup \{p'\|q' \xrightarrow{a} p'\|q'' : p' \xrightarrow{a} p'' \notin \longrightarrow\downarrow S_p,\ q' \xrightarrow{a} q'' \in \longrightarrow\downarrow S_q\}\\
F_{p\|q} &\subseteq \{p'\|q' : p' \in F_p,\ q' \in F_q\}
\end{aligned}$$

Suppose that $s \in L(\mathcal{A}[p\|q])$, *i.e.*, there exists $t \in S_{p\|q} \cap \circledcirc$, such that $p\|q \xrightarrow{s} t$. The proof is done by induction on the length of s. If s is ϵ, t must be $p\|q$ and, therefore, both $p \in \circledcirc$ and $q \in \circledcirc$. Thus, $p \xrightarrow{\epsilon} p$ and $q \xrightarrow{\epsilon} q$, *i.e.*, $\epsilon \in L(\mathcal{A}[p]) \cap L(\mathcal{A}[q])$. Hence, $\epsilon \in L(\mathcal{A}[p])^{\alpha(p)}\|^{\alpha(q)}L(\mathcal{A}[q])$.
If s is as', we have $p\|q \xrightarrow{a} r$, where r can be: (i) $p'\|q'$, assuming that $p \xrightarrow{a} p'$ and $q \xrightarrow{a} q'$; (ii) $p'\|q$, assuming that $p \xrightarrow{a} p'$ and $a \notin \alpha(q)$; (iii)$p\|q'$, assuming that $q \xrightarrow{a} q'$ and $a \notin \alpha(p)$. We are just going to prove (i). The remainder proofs are similar. Since $p\|q \xrightarrow{a} p'\|q'$, we conclude that there exists $t_p\|t_q$, such that $p'\|q' \xrightarrow{s'} t_p\|t_q$. Therefore, $s' \in L(\mathcal{A}[p'\|q'])$. By induction, $s' \in L(\mathcal{A}[p'])^{\alpha(p')}\|^{\alpha(q')}L(\mathcal{A}[q'])$, and thus, there exists $t'_p \in \circledcirc$ and $t'_q \in \circledcirc$, such that $p' \xrightarrow{s'\downarrow\alpha(p')} t'_p$ and $q' \xrightarrow{s'\downarrow\alpha(q')} t'_q$. By applying the rule of Proposition 24, we have $p \xrightarrow{(as')\downarrow\alpha(p')} t'_p$ and $q \xrightarrow{(as')\downarrow\alpha(q')} t'_q$. Therefore, since $\alpha(p) = \alpha(p')$ and $\alpha(q) = \alpha(q')$, we conclude that $s \downarrow \alpha(p) \in L(\mathcal{A}[p])$ and $s \downarrow \alpha(q) \in L(\mathcal{A}[q])$.
Conversely, suppose that $s \in L(\mathcal{A}[p])^{\alpha(p)}\|^{\alpha(q)}L(\mathcal{A}[q])$. The proof is done by induction on the length of s. There are three distinct situations: (a) $a \in L(\mathcal{A}[p]) \cap L(\mathcal{A}[q])$; (b) $a \in L(\mathcal{A}[p])\backslash L(\mathcal{A}[q])$, and (c) $a \in L(\mathcal{A}[q])\backslash L(\mathcal{A}[q])$. We are just going to prove (a). The remainder proofs are similar. If s is ϵ, then $\epsilon \in L(\mathcal{A}[p])$ and $\epsilon \in L(\mathcal{A}[q])$. Thus, $p \xrightarrow{\epsilon} p$ and $q \xrightarrow{\epsilon} q$, with $p, q \in \circledcirc$. Hence, $p\|q \in \circledcirc$ and, consequently, $\epsilon \in L(\mathcal{A}[p\|q])$. If s is as', then $(as') \downarrow \alpha(p) \in L(\mathcal{A}[p])$ and $(as') \downarrow \alpha(q) \in L(\mathcal{A}[q])$, *i.e.*, there exists $t_p \in F_p$ and $t_q \in F_q$, such that $p \xrightarrow{(as')\downarrow\alpha(p')} t_p$ and $q \xrightarrow{as'\downarrow\alpha(q)} t_q$. Assuming now that $p \xrightarrow{a} p'$ and $q \xrightarrow{a} q'$, we conclude that $p' \xrightarrow{s'\downarrow\alpha(p)} t_p$ and $q' \xrightarrow{s'\downarrow\alpha(q)} t_q$, *i.e.*, $s' \in L(\mathcal{A}[p']^{\alpha(p)}\|^{\alpha(q)}L(\mathcal{A}[q'])$. By induction, $s' \in L(\mathcal{A}[p'\|q'])$, i.e., there exists $t'_p, t'_q \in \circledcirc$, such that $p'\|q' \xrightarrow{s'} t'_p\|t'_q$, and applying the rule of Proposition 24, we conclude $p\|q \xrightarrow{as'} t'_p\|t'_q$. Hence, $s \in L(\mathcal{A}[p\|q])$. □

Proof (*Proposition 41*). Given the alphabet U, such that $a \in U$, we have that

(a) $\emptyset_{[a]} = \{s \in \emptyset : \exists s\prime, s'' \in U^*\ s = s'\,a\,s''\} = \emptyset$
(b) $\{a\}_{[a]} = \{s \in \{a\} : \exists s', s'' \in U^*\ s = s'\,a\,s''\} = \{a\}$, where both s' and s'' are ϵ.
(c) Given $b \in U$, such that a and b do not coincide, we have that

$$\{b\}_{[a]} = \{s \in \{b\} : \exists s', s'' \in U^*\ s = s'\,a\,s''\} = \emptyset$$

(d) The proof that $(a.L)_{[a]} \subseteq a.L$ is trivial. Suppose that $s \in a.L$, *i.e.*, s is $a\,s''$, with $s'' \in L$. Considering that s' is ϵ, we conclude that there exists $s', s'' \in U^*$, such that s is $s'\,a\,s''$.
(e) Suppose that $s \in (b.L)_{[a]}$, *i.e.*, there exists $s', s'' \in U^*$, such that s is $b\,s'\,a\,s''$ and $s'\,a\,s'' \in L$. Since $s'\,a\,s'' \in L_{[a]}$, we also have that $s \in b.L_{[a]}$. Conversely, suppose that $s \in b.L_{[a]}$, *i.e.*, s is $b\,t$, with $t \in L_{[a]}$. There exists $s', s'' \in U^*$, such that t is $s'\,a\,s''$, and since $t \in L$, we conclude $s \in (b.L)_{[a]}$.
(f) Given $L, L' \subseteq U^*$, we have that

$$\begin{aligned}
(L \cup L')_{[a]} &= \{s \in L \cup L' : \exists s', s'' \in U^*\ s = s'\,a\,s''\}\\
&= \{s \in L : \exists s', s'' \in U^*\ s = s'as''\} \cup \{s \in L' : \exists s', s'' \in U^*\ s = s'a\,s''\}\\
&= L_{[a]} \cup L'_{[a]}[-1pc]
\end{aligned}$$

(g) Suppose that $L \subseteq T$ and $s \in L_{[a]}$. There exists $s', s'' \in U^*$, such that s is $s'\, a\, s''$. Since $s \in L \subseteq T$, we conclude that $s \in T_{[a]}$. □

Proof (*Proposition 46*). We use induction on the structure of p for the finite terms. We are just going to prove the case where p is $[a]p'$. The remainder proofs are similar to the proofs for the QS model. Assuming, by induction, that there exists $\pi' \in cfProc^{\ddagger}_{QS_{box}}(X)$, such that $\vdash_{QS_{box}} p' = \pi'$, we have that $\vdash_{QS_{box}} [a]p' = [a]\pi'$. The proof follows now by induction on the structure of π', where we assume that a and b do not coincide. Let $U = (\pi')$.

1. If π' is $abort_U$,

$$\vdash_{QS_{box}} [a]abort_U = abort_U \qquad (GL_{01})$$

where $abort_U$ is a term in $cfProc^{\ddagger}_{QS_{box}}(X)$.

2. If π' is $stop_U$,

$$\vdash_{QS_{box}} [a]stop_U = abort_U \qquad (GL_{02})$$

where $abort_U$ is a term in $cfProc^{\ddagger}_{QS_{box}}(X)$.

3. If π' is $!a.\pi''$,

$$\vdash_{QS_{box}} [a]!a.\pi'' = !a.\pi'' \qquad (GL_{03})$$

where $!a.\pi''$ is a term in $cfProc^{\ddagger}_{QS_{box}}(X)$.

4. If π' is $!b.\pi''$, assuming by induction, that there exists $\chi \in cfProc^{\ddagger}_{QS_{box}}(X)$, such that $\vdash_{QS_{box}} [a]\pi'' = \chi$,

$$\begin{aligned} \vdash_{QS_{box}} [a]!b.\pi'' &= !b.[a]\pi'' && (GL_{04}) \\ &= !b.\chi && \text{(induction)} \end{aligned}$$

where $!b.\chi$ is a term in $cfProc^{\ddagger}_{QS_{box}}(X)$.

5. If π' is $a.\pi''$,

$$\begin{aligned} \vdash_{QS_{box}} [a]a.\pi'' &= [a](!a.\pi'' + stop_{\alpha(\pi'')}) && \text{(Liv)} \\ &= [a]!a.\pi'' + [a]stop_{\alpha(\pi'')} && (GL_{05}) \\ &= [a]!a.\pi'' + abort_{\alpha(\pi'')} && (GL_{03}, GL_{01}) \\ &= !a.\pi'' && (S_{04}) \end{aligned}$$

where $!a.\pi''$ is a term in $cfProc^{\ddagger}_{QS_{box}}(X)$.

6. If π' is $b.\pi''$, assuming by induction that there exists $\chi \in cfProc^{\ddagger}_{QS_{box}}(X)$, such that $\vdash_{QS_{box}} [a]\pi'' = \chi$,

$$\begin{aligned} \vdash_{QS_{box}} [a]b.\pi'' &= [a](!b.\pi'' + stop_{\alpha(\pi'')}) && \text{(Liv)} \\ &= [a]!b.\pi'' + [a]stop_{\alpha(\pi'')} && (GL_{05}) \\ &= !b[a].\pi'' + abort_{\alpha(\pi'')} && (GL_{04}, GL_{01}) \\ &= !b[a]\pi'' && (S_{04}) \\ &= !b.\chi && \text{(induction)} \end{aligned}$$

where $!b.\chi$ is a term in $cfProc^{\ddagger}_{QS_{box}}(X)$.

7. If π' is $\pi'_1 + \pi'_2$, assuming by induction that there exists $\chi_1, \chi_2 \in cfProc^{\ddagger}_{QS_{box}}(X)$, such that $\vdash_{QS_{box}} [a]\pi'_1 = \chi_1$ e $\vdash_{QS_{box}} [a]\pi'_2 = \chi_2$,

$$\begin{aligned} \vdash_{QS_{box}} [a](\pi'_1 + \pi'_2) &= [a]\pi'_1 + [a]\pi'_2 && (GL_{05}) \\ &= \chi_1 + \chi_2 && \text{(induction and substitution)} \end{aligned}$$

where $\chi_1 + \chi_2$ is a term in $cfProc^{\ddagger}_{QS_{box}}(X)$.

The prove for recursive terms follows from ω-induction rule. □

Proof (*Proposition 51*). Let $a, b \in Act$, with $a \neq b$. Given $p \in cProc'_{QS_{box}}(X)$, such that $\mathcal{A}_{box}[p] = \langle S_p, p, \alpha(p), \longrightarrow_{box} \downarrow S_p, F_p \rangle$, we have that $\mathcal{A}_{box}[[a]p] = \langle S_{[a]p}, [a]p, \alpha([a]p), \longrightarrow_{box} \downarrow S_{[a]p}, F_{[a]p} \rangle$, where: (i) $S_{[a]p} \subseteq \{[a]p' : p' \in S_p\} \cup S_p$ and also $\#S_{[a]p} = \#S_p$, (ii) $\alpha([a]p) = \alpha(p)$, (iii) $\longrightarrow_{box} \downarrow S_{[a]p} \subseteq \{[a]p' \xrightarrow{a}_{box} p'' : p' \xrightarrow{a}_{box} p'' \in \longrightarrow_{box} \downarrow S_p\} \cup \{[a]p' \xrightarrow{b}_{box} [a]p'' : p' \xrightarrow{b}_{box} p'' \in \longrightarrow_{box} \downarrow S_p\} \cup \longrightarrow_{box} \downarrow S_p$ and (iv) $F_{[a]p} \subseteq F_p$. Suppose that $s \in L(\mathcal{A}_{box}[[a]p])$, *i.e.*, there exists $t \in F_{[a]p}$, such that $[a]p \xrightarrow{s}{}^*_{box} t$. Since $[a]p \in \bigcirc$, s can not be ϵ, so it follows that s can be either $a\, s'$ or $b\, s'$, with $s' \in \alpha(p)^*$. The proof follows by induction on the length of s. Conversely, suppose that $s \in L(\mathcal{A}_{box}[p])_{[a]}$, *i.e.*, there exists $t \in F_p$ such that $p \xrightarrow{s}{}^*_{box} t$ and s contains a. We use again induction on the length of s. □

Proof (*Proposition 56*). We use induction on the length of derivation of theorem $\vdash_{QS_{\text{box}}} p \leq q$, after proving the soundness of the system $dc(\mathcal{QS}_{\text{box}})$. We use the Proposition 41 to prove the soundness of the global liveness proper axioms.
Soundness of GL_{01}.

$$\begin{aligned}
\mathbb{QS}_{\text{box}}[\![[a]abort_U]\!]\rho_{QS_{\text{box}}} &=_{QS_{\text{box}}} [a]_{QS_{\text{box}}}\mathbb{QS}_{\text{box}}[\![abort_U]\!]\rho_{QS_{\text{box}}}\\
&=_{QS_{\text{box}}} [a]_{QS_{\text{box}}}\langle U,\emptyset\rangle\\
&=_{QS_{\text{box}}} \langle U,\emptyset\rangle\\
&=_{QS_{\text{box}}} \mathbb{QS}_{\text{box}}[\![abort_U]\!]\rho_{QS_{\text{box}}}
\end{aligned}$$

Soundness of GL_{02}.

$$\begin{aligned}
\mathbb{QS}_{\text{box}}[\![[a]stop_U]\!]\rho_{QS_{\text{box}}} &=_{QS_{\text{box}}} [a]_{QS_{\text{box}}}\mathbb{QS}_{\text{box}}[\![stop_U]\!]\rho_{QS_{\text{box}}}\\
&=_{QS_{\text{box}}} [a]_{QS_{\text{box}}}\langle U,\{\epsilon\}\rangle\\
&=_{QS_{\text{box}}} \langle U,\emptyset\rangle\\
&=_{QS_{\text{box}}} \mathbb{QS}_{\text{box}}[\![abort_U]\!]\rho_{QS_{\text{box}}}
\end{aligned}$$

Soundness of GL_{03}.

$$\begin{aligned}
\mathbb{QS}_{\text{box}}[\![[a]!a.x_U]\!]\rho_{QS_{\text{box}}} &=_{QS_{\text{box}}} [a]_{QS_{\text{box}}}\mathbb{QS}_{\text{box}}[\![!a.x_U]\!]\rho_{QS_{\text{box}}}\\
&=_{QS_{\text{box}}} [a]_{QS_{\text{box}}}\langle U, a.\tau(\mathbb{QS}_{\text{box}}[\![x_U]\!]\rho_{QS_{\text{box}}})\rangle\\
&=_{QS_{\text{box}}} \langle U\cup\{a\}, a.\tau(\mathbb{QS}_{\text{box}}[\![x_U]\!]\rho_{QS_{\text{box}}})\rangle\\
&=_{QS_{\text{box}}} \mathbb{QS}_{\text{box}}[\![!a.x_U]\!]\rho_{QS_{\text{box}}}
\end{aligned}$$

Soundness of GL_{04}.

$$\begin{aligned}
\mathbb{QS}_{\text{box}}[\![[a]!b.x_U]\!]\rho_{QS_{\text{box}}} &=_{QS_{\text{box}}} [a]_{QS_{\text{box}}}\mathbb{QS}_{\text{box}}[\![!b.x_U]\!]\rho_{QS_{\text{box}}}\\
&=_{QS_{\text{box}}} [a]_{QS_{\text{box}}}\langle U, b.\tau(\mathbb{QS}_{\text{box}}[\![x_U]\!]\rho_{QS_{\text{box}}})\rangle\\
&=_{QS_{\text{box}}} \langle U\cup\{a\}, b.(\tau(\mathbb{QS}_{\text{box}}[\![x_U]\!]\rho_{QS_{\text{box}}}))_{[}a]\rangle\\
&=_{QS_{\text{box}}} !b._{QS_{\text{box}}}\langle U\cup\{a\}, (\tau(\mathbb{QS}_{\text{box}}[\![x_U]\!]\rho_{QS_{\text{box}}}))_{[}a]\rangle\\
&=_{QS_{\text{box}}} !b._{QS_{\text{box}}}\mathbb{QS}_{\text{box}}[\![[a]x_U]\!]\rho_{QS_{\text{box}}}\\
&=_{QS_{\text{box}}} \mathbb{QS}_{\text{box}}[\![!b.[a]x_U]\!]\rho_{QS_{\text{box}}}
\end{aligned}$$

Soundness of GL_{05}.

$$\begin{aligned}
\mathbb{QS}_{\text{box}}[\![[a](x_U+y_U)]\!]\rho_{QS_{\text{box}}} &=_{QS_{\text{box}}} [a]_{QS_{\text{box}}}\mathbb{QS}_{\text{box}}[\![x_U+y_U]\!]\rho_{QS_{\text{box}}}\\
&=_{QS_{\text{box}}} [a]_{QS_{\text{box}}}\langle U,\tau(\mathbb{QS}_{\text{box}}[\![x_U]\!]\rho_{QS_{\text{box}}})\cup\tau(\mathbb{QS}_{\text{box}}[\![y_U]\!]\rho_{QS_{\text{box}}})\rangle\\
&=_{QS_{\text{box}}} \langle U\cup\{a\}, (\tau(\mathbb{QS}_{\text{box}}[\![x_U]\!]\rho_{QS_{\text{box}}})\cup\tau(\mathbb{QS}_{\text{box}}[\![y_U]\!]\rho_{QS_{\text{box}}}))_{[}a]\rangle\\
&=_{QS_{\text{box}}} \langle U\cup\{a\}, (\tau(\mathbb{QS}_{\text{box}}[\![x_U]\!]\rho_{QS_{\text{box}}}))_{[}a]\cup(\tau(\mathbb{QS}_{\text{box}}[\![y_U]\!]\rho_{QS_{\text{box}}}))_{[}a]\rangle\\
&=_{QS_{\text{box}}} \mathbb{QS}_{\text{box}}[\![[a]x_U]\!]\rho_{QS_{\text{box}}} +_{QS_{\text{box}}} \mathbb{QS}_{\text{box}}[\![[a]y_U]\!]\rho_{QS_{\text{box}}}\\
&=_{QS_{\text{box}}} \mathbb{QS}_{\text{box}}[\![[a]x_U+[a]y_U]\!]\rho_{QS_{\text{box}}}
\end{aligned}$$

References

[1] L. Aceto, W. Fokkink, C. Verhoef, Conservative extension in structural operational semantics, in: Current Trends in Theoretical Computer Science, 2001, pp. 504–524.
[2] J.C.M., Baeten, A brief history of process algebra, Tech. Rep. CS-R 04/02, Dept. of Comput. Sci., Technische Universiteit Eindhoven, 2004.
[3] J.C.M. Baeten, W.P. Weijland, Process algebra, vol. 18, Cambridge University Press, 1990
[4] M. Broy, E.-R. Olderog, Handbook of Process Algebra, Elsevier Science Publishers B.V., 2001, pp. 101–195 (Chapter 2).
[5] J.F. Costa, Teoria algébrica dos processos animados, Master's thesis, Universidade Técnica de Lisboa, 1989.
[6] J.F. Costa, Fundamentos matemáticos da concorrência, Ph.D. thesis, Universidade Técnica de Lisboa, 1991.
[7] J.F. Costa, A. Sernadas, Progress assumption in concurrent systems, Formal Aspects Comput. 7 (1) (1995) 18–36.
[8] H. Ehrig, B. Mahr, Monographs on Theoretical Computer Science, Springer-Verlag, 1985.
[9] R. Gorrieri, U. Montanari, Towards hierarchical specification of systems: a proof system for strong prefixing, Int. J. Found. Comput. Sci. 1 (3) (1990) 277–293.
[10] H. Guerra, J.F. Costa, Processes with infinite liveness requirements, in preparation.
[11] M. Hennessy, Algebraic Theory of Processes, MIT Press, 1988.
[12] C.A.R. Hoare, Communicating Sequential Processes, Prentice-Hall, 1985.
[13] J. Hopcroft, R. Motwani, J. Ullman, Languages and Computation, second ed., Addison Wesley, 2001.
[14] B. Jonsson, A model and proof system for asynchronous networks, in: Proceedings of the 4th Annual ACM Symposium on Principles on Distributed Computing, 1985, pp. 49–58.

[15] B. Jonsson, A fully abstract trace model for dataflow networks, in: Proceedings of the 16th Annual Symposium on Principles of Programming Languages, 1989, pp. 155–165.
[16] N.A., Lynch, M.R., Tuttle, Hierarchial correctness proofs for distributed algorithms, in: Proceedings of the 6th Annual ACM Symposium on Principles of Distributed Computing, ACM, 1987, pp. 137–151.
[17] N.A. Lynch, M.R. Tuttle, Introduction to input/output automata 2 (3) (1989) 219–246.
[18] Z. Manna, A. Pnueli, The Temporal Logic of Reactive and Concurrent Systems – Specification, Springer-Verlag, 1992.
[19] R. Milner, Communication and Concurrency, Prentice Hall, 1989.
[20] R. Milner, Operational and Algebraic Semantics of Concurrent Systems, Handbook of Theoretical Computer Science, Elsevier Science Publishers B.V., 1990, pp. 1203–1242 (Chapter 19).
[21] J. Misra, Reasoning about networks of communicating processes, in: INRIA Advanced Nato Study Institute on Logics and Models for Verification and Specification of Concurrent Systems, 1984.
[22] J. Misra, K. Chandy, Proofs of networks of processes, IEEE Trans. Softw. Eng. vol. 1 (1981) 417–426.
[23] J. Misra, K.T.S. Chandy, Proving safety and liveness of communicating processes with examples, in: Proceedings of the ACM SIGACT-SGOPS Symposium on Principles of Distributed Computing, 1982, pp. 201–208.
[24] E. Olderog, Nets, Terms and Formulas, Cambridge University Press, 1991.
[25] E.R. Olderog, C.A.R. Hoare, Specification-oriented semantics for communicating processes, Acta Inform. (23) (1986) 9–66.
[26] S. Owicki, L. Lamport, Proving Liveness Properties of Concurrent Systems, ACM Transactions on Programming Languages and Systems, ACM, 1982, pp. 155–495.
[27] G.D. Plotkin, A structural approach to operational semantics, Tech. Rep. FN-19, Computer Science Dept., Universtity of Aarhus, 1981.
[28] G.D. Plotkin, The origins of structural operational semantics, J. Logic Algebraic Program. (2004) (Special Issue on Structural Operational Semantics).
[29] A.W. Roscoe, The Theory and Practice of Concurrency, Prentice Hall, 1998.
[30] A. Sernadas, H.-D. Ehrich, J.F. Costa, From processes to objects, INESC J. Res. Develop. 1 (1) (1990) 7–27.
[31] V. Stoltenberg-Hansen, I. Lindstrm, E.R. Griffor, Mathematical Theory of Domains, Cambridge University Press, 1994.
[32] W. Thomas, Automata on Infinite Objects, Handbook of Theoretical Computer Science, Elsevier Science Publishers B.V., 1990, pp. 135–191 (Chapter 4).
[33] C. Verhoef, A general conservative extension theorem in process algebras with inequalities, Theoret. Comput. Sci. 2 (177) (1997) 351–380.